Universidad de las Ciencias Informáticas

Facultad 8

Título: Análisis y Diseño del Sistema de Gestión de la Calidad de la Facultad 8.Módulo Pruebas

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autor: Leovys Erix Salazar Suzeta

Tutora: Ing. Greisy Gálvez George

Co Tutora: Ing. Lisset Rosas Moreno

Ciudad de la Habana, Junio 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

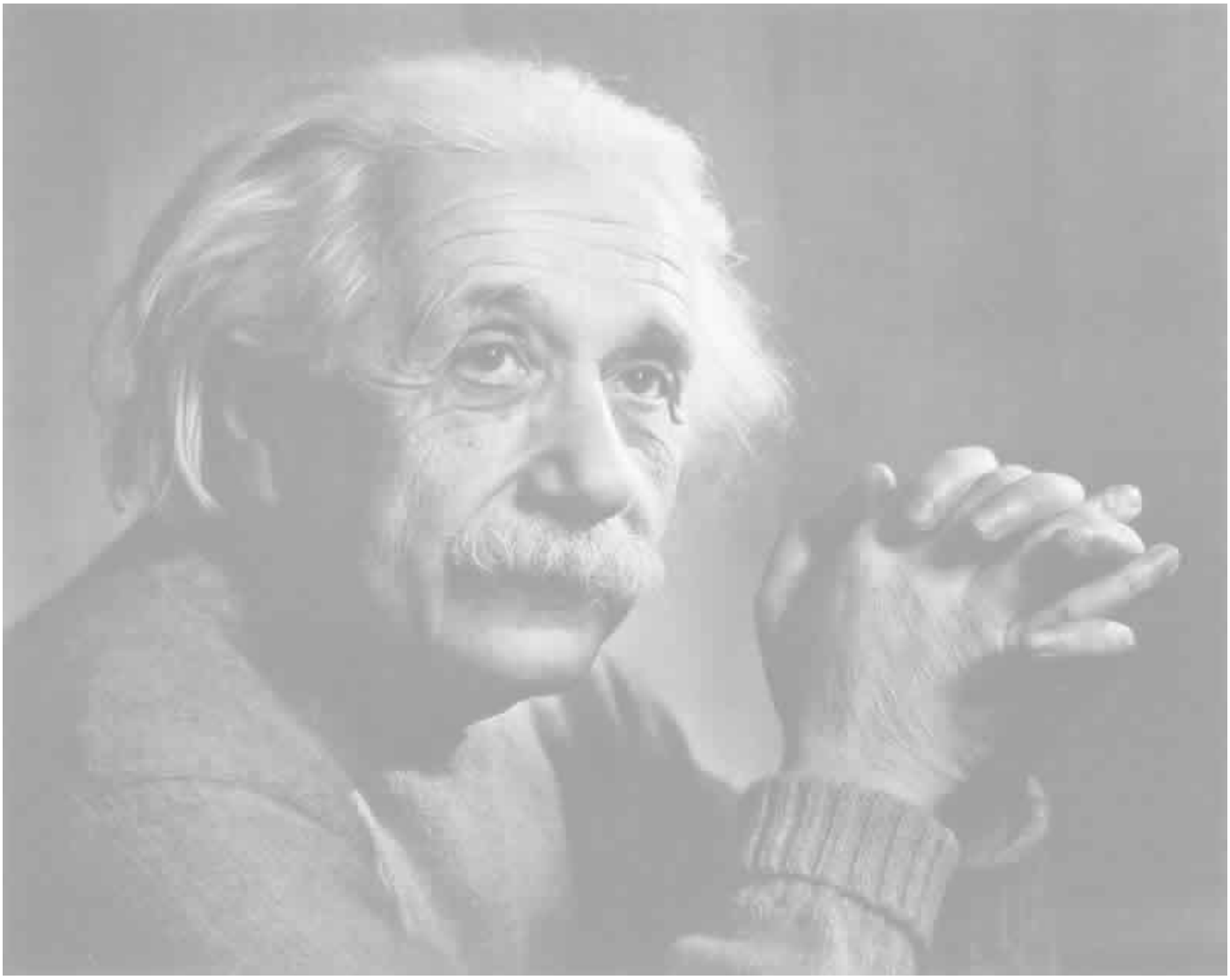
DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste firmo la presente a los _____días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Tutor
Leovys Erix Salazar Suzeta	Ing. Greisy Gálvez George

Pensamiento



"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

Agradecimientos

A mis compañeros tesistas Andrés, Liniet, Roberto y Emilio por su esfuerzo y ayuda en este trabajo.

A mis compañeros de aula que tanto cariño se desborda de sus corazones.

A los tutores Greisy y Lisset por tanto ahínco, tesón y la dedicación con que asumieron en este trabajo.

A todo el tribunal por su dedicación en el transcurso de la tesis, Dixson, Mailen, Celia, Yunexis.

Al oponente Ramón por estar al tanto y tener tiempo para las dudas que me surgían.

A los compañeros y amigos del Proyecto de Calidad Yamila, Annia, Camejo, Leonel, Wisel, Yalida, las dos Lisbet, Yoan, Yoerlis (Pico), Julio, Lisay, Ariel, Arianna, Mariné, Indira, Pedro, Lizandra, Alice, Yadira, Dariel, Josefina, Yaneisy, a las Técnicas.

A mis profesores de toda la carrera Socarras, Aliuska, Dosagues, Mijail, Yasiris, Yasim, Arlenis, Tomás, Surelis, Sergio, Héctor, Harold, y Heiser.

A mis compañeros de apartamento Alexei (El Salvaje), Héctor, Yoisel (El Mío), Rodolkis, Ariam, Román, Abdiel, Boris (El Chino).

A todos los amigos y amigas de la Facultad 8, así como sus profesores y trabajadores.

A los mulos de la facultad 8 Michel, Lautaro, Yasser, Yoisel, Lucas, Andrés, Roli, Rodolki, El Pombo, Ronny, Leonardo, Roelvis, Yudiel, Ariel, Karel, Abdiel, Yudito, Ariam, Marlon, Pico, Yudislandry, Henry, Nelson, Yunior, Moreno, Ernesto, Héctor.

A todos los compañeros y amigos que no culminaron los estudios Suyen, Hennry, Lisandra, Yolanda, Elizabeth, Eneida.

A Rolian (Roli), Lautaro, (Lapto), Andrés compañeros y amigos de toda la carrera.

A todas las muchachas hermosas de mi aula Yaneisy, Elizabeth, Mairelis, Marcia, Maité, Beatris, Dalaity, Josefina, Yadira, Doris.

A mi Revolución que tanto ha hecho por los hombres del futuro.

Al Comandante en Jefe Fidel Castro líder indiscutible de la Revolución Cubana.

A la UCI por habernos formado como profesionales y hacerme con su andar hombre de bien.

Dedicatoria

A mi mamá Damaris, mi papá Ernio y mis dos hermanos Erlis y Ronald que tanto amor y cariño depositaron en mí y son los mayores impulsores de este trabajo.

A mi abuela Caridad que no se encuentra entre nosotros pero siempre me dio aliento.

A todos mis familiares.

A mis primos Yoirdan (Yordi), Lazarito, Armandito, Reidy.

A mis primas Yamina, Yamilé, Daneisy, Yaneisy, Yamilka

A mi abuela Maximina que siempre estuvo al tanto de mis logros.

A todos mis familiares que ya no se encuentran.

A mis tíos Papo, Luis, Armando, Pancho, Conrrado, Rey, Aleixys, Edy, Ñico.

A mis tías Deisy, Damaris (Mima), Vitorina, Zenaida.

A mi vecinos queridos Nemesia, Ermes, Yaimara (Tata) Yisel, Ermeduardo (Pucho), Puchungo, Norma, Yanay, Morfo, Caridad, Angelito, Aylin, Chicho, Rosa, Gorito, Elder, Lexter, René, Chinea y demás vecinos.

A mis amigos de El Salvador Yoelvis, Armando, Esnaider, Arolde, Iván, Yoannel, Raúl, Ledis, Robert, Heydrich (Pompo), Maikel, Yudiel, Ángel y Pico.

A mis queridos amigos en la casa de Cuca, Marlenes, Ñiquito, Lisanne, Susanne, Eidol, Ñico.

Índice de Contenido

Índice de Tabla

Índice de Figuras

Resumen

La Universidad de las Ciencias Informáticas (UCI) trabaja para que sus productos de software tengan la calidad requerida y cada facultad tiene como objetivo obtener proyectos con calidad. En la facultad 8 se cuenta con un grupo de estudiantes que son los encargados de realizar las revisiones de la documentación y los diferentes tipos de prueba al software, estas tareas se realizan con un alto grado de trabajo porque no se cuenta con un sistema que se capaz de controlar la calidad requerida por parte de los proyectos. El presente trabajo tiene como objetivo analizar y diseñar un sistema de Gestión de la Calidad en la Facultad 8 que permita al proyecto de calidad posibilitar el proceso de pruebas para garantizar la gestión de la calidad de los proyectos productivos. Para la realización de la propuesta se utilizaron: PHP como lenguaje de programación, PostgreSQL como gestor de base de datos (BD), Apache como servidor Web, Rational Unified Process (RUP) como metodología de desarrollo, Visual Paradigm como herramienta de modelado y Lenguaje Unificado de Modelado (UML) como lenguaje de modelado de sistemas de software que permitió la creación de un conjunto de artefactos que facilitarán la implementación exitosa del sistema.

Introducción

La creación de sistemas informáticos en el mundo actual tiene gran auge, desarrollarlos en el tiempo establecido y con la calidad requerida es una meta de todo equipo de proyecto. Existen varios inconvenientes en el desarrollo de un sistema: mal funcionamiento, documentación con errores, mala planificación de las fases de desarrollo, productos finales que no siguen las normas establecidas; aspectos que como otros, influyen directamente en la calidad y aceptación esperada por parte del cliente.

La Universidad de las Ciencias Informáticas (UCI) creada en septiembre del 2002, se compone de 10 facultades. Este centro de altos estudios, constituye una estrategia en la informatización del país para elevar la cultura informática, la exportación de software y la cultura general integral. Todas las facultades están vinculadas al proceso productivo, donde cada una cuenta con un proyecto que garantiza y vela por la calidad de los productos que desarrollan, su objetivo es gestionar la calidad en los proyectos comprendidos dentro de su área.

La Facultad 8 cuenta con su respectivo Proyecto de Calidad. El Proyecto asume diversas actividades orientando, guiando y monitoreando los procesos de calidad en las fases de desarrollo de cada proyecto productivo, siendo más frecuente las relacionadas con la liberación de productos terminados. En tal sentido, la realización de pruebas se convierte en un proceso fundamental dentro de la vida del Proyecto.

Sin embargo, no se ha logrado una óptima gestión de las anomalías que se generan en cada iteración, se producen errores en cuanto al procesamiento de la información por la extensa documentación existente, la información necesaria para la realización de las pruebas no se encuentra centralizada, no existe la suficiente motivación del personal para realizar las pruebas. Existe dificultad con el llenado del control de versiones, lo que hace difícil identificar quienes cometen errores en la documentación que se entrega, impidiendo tomar medidas que impliquen rectificaciones a tiempo. Todas estas dificultades se traducen en atrasos que repercuten negativamente en el cumplimiento de los cronogramas y liberación de dichos productos.

Por las razones antes expuestas, y otras dificultades relacionadas a las actividades de Calidad que se realizan en la Facultad 8, la dirección del Proyecto de Calidad identifica la necesidad de implementar un Sistema de Gestión de la Calidad, que automatice los principales procesos y contribuya a optimizar el control de actividades tan fundamentales como las pruebas, en las cuales se detectan errores en la

documentación, en el funcionamiento del sistema, en la interfaz de usuario y se garantiza el cumplimiento de requisitos definidos, logrando la satisfacción del cliente.

De lo anterior puede plantearse el **Problema a resolver**:

¿Cómo organizar el proceso de pruebas del proyecto de Calidad de la Facultad 8?

Aportes prácticos esperados del trabajo.

A partir del problema planteado se espera que con la realización de este trabajo se tengan los elementos necesarios para la implementación y puesta en marcha del sistema que entre otras funciones, facilite las actividades relacionadas con las pruebas.

El **objeto de estudio** lo conforman los procesos de pruebas, definiéndose como **campo de acción,** aquellas actividades automatizables dentro de los procesos de pruebas que realiza el Proyecto de Calidad de la Facultad 8. Como **idea a defender** partimos de, que si se realiza el análisis y diseño del módulo pruebas, se podrá contar con un sistema que entre sus funcionalidades incluya la gestión de los procesos de pruebas en la Facultad 8.

Para resolver el problema se trazó como **objetivo genera**l: Realizar el análisis y diseño del módulo Pruebas para el Sistema de Gestión de la Calidad de la Facultad 8.

Los siguientes **objetivos específicos** conducen el desarrollo de la presente investigación:

1. Investigar sobre los procesos de pruebas de manera general y en la Facultad 8.
2. Realizar la ingeniería de requerimientos para el módulo Pruebas.

3. Realizar el análisis y diseño del módulo Pruebas.

Para el cumplimiento de los objetivos se definieron las siguientes **tareas:**

1. Analizar los procesos de prueba que se realizan en la Facultad 8.
2. Representar el negocio.
3. Definir los requisitos del sistema.
4. Identificar las actividades automatizables.
5. Generar artefactos referentes al flujo de trabajo de análisis.
6. Generar artefactos referentes al flujo de trabajo de diseño.

Introducción

Actualidad y necesidad del trabajo.

Las mejoras de procesos a través de herramientas viabilizan el camino hacia productos más cercanos a la excelencia. El perfeccionamiento interno es de gran importancia para cualquier organización que opte por un nivel de calidad apreciable. La dirección del Proyecto de Calidad de la facultad 8 pretende unirse en ese sentido, de ahí el imperativo de desarrollar un sistema que sea capaz de gestionar la calidad en el ámbito de sus proyectos productivos.
La investigación está distribuida en tres capítulos, en los cuales se tratan los siguientes aspectos:

Capítulo I. Incluye un estado del arte de la gestión de la calidad, a nivel internacional, nacional y de la Universidad, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad sobre la gestión de la calidad.

Capítulo II. Posee las características principales del sistema, especificándose en el objeto de estudio con su problema y situación problémica mediante la descripción de los oobjetivos estratégicos de la organización y procesos de negocio que los soportan, el flujo actual de los eventos y el análisis crítico de cómo se ejecutan actualmente esos procesos. También se muestra el objeto de automatización describiendo los procesos que son objetos de automatización y los sistemas automatizados que existen. Se definen los requisitos funcionales y no funcionales del sistema. Se presenta la información que se maneja, definiendo el modelo de negocio representando los actores y trabajadores del negocio, y el diagrama de caso de uso del negocio.

Capítulo III. En este capítulo se presentan los diagramas de clases del análisis, los diagramas de secuencia de los casos de usos arquitectónicamente significativos, los diagramas de clases del diseño con estereotipos web, cada clase posee una breve descripción Se define el ddiagrama de entidad relación de la Base de Datos (BD) y se describen las tablas. También se presentan las definiciones de diseño que se aplican, el tratamiento de errores, la seguridad, la interfaz y la concepción de la ayuda.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

El presente capitulo expone los basamentos teóricos del trabajo realizado ,elaborándose un estudio de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad y teniendo en cuenta las características estudiadas , se adoptan los elementos a utilizar para darle seguimiento a la investigación.

1.2 Tendencias en la actualidad

La creación de sistemas informáticos con calidad es una de las tareas más desafiantes por cualquier equipo de desarrollo u organización que tenga la misión de crear productos de alta calidad para lograr la plena satisfacción del cliente. La tendencia de la calidad comenzó en los años setenta y ochenta llamándose gestión de la calidad total (GTC).

La complejidad de los mercados debido al aumento de la competencia y la facilidad creciente para los envíos o desplazamientos (comercio vía INTERNET, etc.), el surgimiento de nuevas tecnologías y paradigmas empuja hacia la necesidad de la mejora constante, razón esta de desarrollo y puesta en marcha por la empresas de procesos para la gestión y perfeccionamiento de la calidad de sus productos, de forma que asegure a su cliente un pedido con características establecidas.

La calidad requerida de los sistemas de software que se construyen en la Facultad 8 está muy ligada al desarrollo de los procesos de pruebas que se realizan, los cuales constituyen el eje principal para la liberación de productos informáticos logrando una aceptación adecuada por parte de los clientes.

Con lo expuesto se arriba a que el proceso de pruebas proporciona calidad a los productos, productos que deben estar preparados para las mejoras continuas que puedan ser requeridas por los clientes.

1.3 Técnicas

Las técnicas de pruebas de software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Las pruebas de calidad se hacen con la finalidad de descubrir un error y tienen éxito si se descubre un error no detectado hasta entonces, definiéndose un buen caso de prueba aquel que tiene una gran probabilidad de encontrar un error no descubierto hasta entonces. Uno de los objetivos en el flujo de prueba es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. (**PRESSMAN, 2005).**

Para realizar el diseño de casos de prueba se usan dos categorías de técnicas de diseño de casos de prueba: prueba de caja blanca y prueba de caja negra.

Las pruebas de caja blanca se centran en el control de la estructura del programa, según (**Hetzel,1984),** se les llama <<pruebas a pequeñas escala>> por ser aplicadas a pequeños componentes de programas por ejemplo: módulos o pequeños grupos de módulo. Esto no ocurre con las pruebas de caja negra las que se pueden denominar <<pruebas a gran escala>> porque son aplicables para validar los requisitos funcionales. (**PRESSMAN, 2005).**

Vistas estas técnicas se pueden considerar aplicables a los procesos de pruebas llevados por la facultad.

1.4 Software usado en la actualidad

En la actualidad existen muchas herramientas que se dedican a la gestión de la calidad, entre ella se exponen dos de las existentes:

1.4.1 ISODOC®

Versión: 4.5 / Año: 2007 • (Colombia)

ISODOC® Es una solución líder, diseñada para el manejo y gestión de Sistemas de Calidad. ISODOC permite una completa administración sobre la estructura documental según normas ISO, BASC, RI etc. Las funciones principales de ISODOC son: Gerenciamiento y control de la documentación desde su elaboración hasta su aprobación, Auditorías, Acciones Correctivas y Planes de Acción, Gestión de Recursos, Gestión de Procesos, Administración de Proveedores, Ciclo de Mejoramiento del Sistema de Calidad, Control de Copias Impresas e Indicadores de Gestión. El sistema genera alarmas e informes gerenciales especializados según sus necesidades. ISODOC combina elementos claves que garantizan el

éxito: conceptos sólidos sobre el Sistema de Gestión de la Calidad según normas ISO9000 y la aplicación de tecnologías expertas en Administración de documentos, y colaboración. ISODOC permite automatizar los Indicadores de Gestión, e integrarse con nuestros sistemas de Gestión de Recursos Humanos (RHCONTROL) y el módulo de Servicio al Cliente (SQRCONTROL). Todas estas soluciones y módulos pueden implementarse independientemente o de manera conjunta siempre vía Web (en la Intranet de su Empresa o Extranet) obteniendo así un sistema de información completa y eficiente, especializada en Sistemas de Calidad. (**Catalogo, 2007).**

Requerimientos:

SERVIDOR: Windows 2000 / 2003 Server / PRO, LINUX, Solaris, AS / 400.

- 1 GB Libre en Disco Duro.
- 512 RAM.
- Compatible con Bases de datos Oracle / SQL / DB2 / MySQL.
- Ayudas Online.
- Computadores Usuarios: Cualquier computador que pueda correr un navegador de Internet (Explorer, Mozilla, Copérnico, Opera, Netscape). (**Catalogo, 2007).**

1.4.2 VISION EMPRESARIAL® - SGC

Versión: 5.0 / Año: 2008 • Desarrollo: PENSEMOS S.A. (Colombia)

VISION EMPRESARIAL® SGC Es una solución que facilita la implementación, mantenimiento y mejora de su Sistema de Gestión de Calidad, integrando la información clave de los procesos, a través de la gestión documental, el seguimiento a las mejoras, el manejo de las auditorías internas, la ejecución de la revisión gerencial y la gestión de riesgos. (**Catalogo, 2007).**

CARACTERÍSTICAS PRINCIPALES: Organiza y facilita el proceso de revisión gerencial y ejecución de Auditorias de Calidad. Permite configurar el Mapa de procesos de una manera gráfica y caracterizar cada proceso, teniendo acceso además a sus documentos relacionados e indicadores definidos. Optimiza la gestión de la documentación y su normalización, permitiendo la participación activa del personal de la organización. Apoya la mejora continua mediante el seguimiento de quejas y reclamos, acciones correctivas y preventivas, ideas de mejora, control de producto no conforme y seguimiento a los indicadores de los procesos. Utiliza el correo electrónico para informar las novedades sobre los documentos a la lista de distribución y responsables. Los usuarios pueden acceder a través de la Intranet

o Internet ya que es completamente WEB. Hace parte de toda una Suite junto con los Módulos de Balanced Scorecard e Integridad Operativa. (**Catalogo, 2007).**

1.4.3 Según Software usados

Una vez analizados los software ISODOC® y VISION EMPRESARIAL se concluye que ISODOC® no se adapta a las condiciones de la facultad 8 porque está muy basada en la administración sobre la estructura documental según normas ISO, BASC, RI y no se centra en el proceso de Pruebas, a pesar de realizar el proceso de auditorías se busca centrarse en proporcionar mecanismos eficaces de pruebas. Y ISION EMPRESARIAL® - SGC no se corresponde con las condiciones de la facultad porque está muy basado en la gestión documental, el manejo de las auditorías internas, la ejecución de la revisión gerencial y la gestión de riesgos sin centrarse en el proceso de pruebas necesario para minimizar los errores en los proyectos.

Dados estos aspectos se necesita escoger las tecnologías a utilizar para el desarrollo del sistema:

1.5 Tecnologías

A continuación se describen algunas tecnologías actuales posibles a utilizar para obtener una solución óptima a los problemas que se plantearon anteriormente. De acuerdo con las condiciones existentes y las necesidades se explican las tecnologías que pueden guiar al sistema que se va a construir para lograr una mayor eficiencia.

1.5.1 Aplicación de escritorio

Aplicación de escritorio: En la actualidad existen dos plataformas de desarrollo para las aplicaciones, las aplicaciones de escritorio (desktop) y las Web. Las aplicaciones de escritorio son las que se necesitan un tipo de instalación en el cliente, gran parte de estas consumen enormes cantidades de espacio en disco dependiendo del tamaño necesitando grandes cantidades de memoria.

1.5.2 Aplicación web

Aplicaciones web**:** Página web especializada que permite mostrar o captar información en bases de datos. Su arquitectura general es la de un sistema cliente/servidor, donde tanto el cliente (el navegador) como el servidor (el servidor Web), y el protocolo mediante el que se comunican (HTTP) son estándares.

1.5.3 Lenguaje de Programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas, definen su estructura y el significado de sus elementos y expresiones. Es una técnica estándar de comunicación utilizada para controlar el comportamiento físico y lógico de una máquina. Permite expresar las instrucciones que han de ser ejecutadas en una computadora.

1.5.4 PHP 5.0

PHP **H**ypertext **P**re-processor (PHP) es una de las tecnologías Web más extendida en la actualidad, concebido inicialmente para trabajar sobre Linux con servidor Apache, pero hoy en día puede alojarse en cualquier servidor. El código fuente está abierto, por los que los problemas que se presentan son rápidamente controlados, y solucionados; excelente biblioteca de funciones. Posee la capacidad de conexión y soporte con la mayoría de los gestores de bases de datos (MySQL, Oracle, PostgreSql), así como con la mayoría de los servidores Web que se utilizan en la actualidad.

El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática, librándose el usuario de tener que separar las variables y sus valores. Es un lenguaje script, no compilado; un lenguaje de bajo nivel donde dificulta la modularización y organización por capa de la aplicación. La orientación a objeto es deficiente para grandes aplicaciones. Todo el trabajo lo realiza el servidor y no delega al cliente, por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número. La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.Funciona en las plataformas de *Windows, Linux/Unix*, *Solaris* con el software servidor apropiado.

1.5.5 ASP

ASP no es un lenguaje de programación por sí solo. Es mis como el pegamento que mantiene unidos los scripts, objetos, componentes e interacciones con el servidor Web.Ténicamente, ASP se compone de objetos que son llamados desde VBScript o JScript para realizar funciones altamente útiles, como capturar datos enviados por los usuarios, responder a entrada de usuarios, administrar aplicaciones y sesiones y administrar el servidor.

1.5.6 Python

Python es un lenguaje interpretado, interactivo y orientado a objetos. A menudo es comparado con, Perl y Java. Combina una gran potencia con una sintaxis clara (aunque no simplificada más de lo conveniente). Otros de sus puntos fuertes son la portabilidad, facilidad de aprendizaje y la existencia de potentes bibliotecas estándar.

1.5.7 ¿Por qué PHP?

Se decide PHP porque presenta gran velocidad a la hora de procesar los datos, puede operar en la mayoría de los sistemas operativos (GNU/Linux, MacOS, Windows) y otros, por la gran característica de alojarse en cualquier servidor, gestión de memoria automática, rápido, fácil de aprender y por la integración de trabajo con la bases de datos de PostgreSQL. Además que posee programación orientada a objeto.

1.6 Metodologías existentes

Existen diversas metodologías para crear productos de software, unas menos eficientes que otras y menos adaptables a diferentes tipos de proyectos, pero todas ayudan a la guía y construcción de software informáticos, seguido se exponen dos metodologías, las cuales son RUP y XP.

1.6.1 Rational Unified Process

Rational Unified Process: Es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. Es un Proceso Práctico. **(Rational, 2007).**

RUP se caracteriza por ser:

1. Iterativo e incremental
2. Dirigido por caso de uso
3. Centrado en la arquitectura

RUP posee cuatro fases:

1. Inicio
2. Elaboración

3. Construcción
4. Transición

Las mejores prácticas del **Rational Unified Process**, (**RUP**), son un conjunto de procesos web-enabled de ingeniería de software que dan guía para conducir las actividades de desarrollo del equipo. Como una plataforma de procesos que abarca todas las prácticas de la industria, el **RUP** permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Se podrán alcanzar resultados predecibles unificando el equipo con procesos comunes que optimicen la comunicación y creen un entendimiento común para todas las tareas, responsabilidades y artefactos. Desde un único sitio web centralizado de intercambio, el Software Rational, las plataformas, herramientas y expertos de dominios proveen los componentes de proceso necesarios para el éxito. **(Rational, 2007).**

El **Rational Unified Process** unifica todo el equipo de desarrollo de software y optimiza su comunicación proveyendo a cada miembro de una aproximación al desarrollo de software con una base de conocimiento on-line customizable de acuerdo a las necesidades específicas del proyecto. Usando la navegación on-line del browser, cada miembro del equipo tiene acceso instantáneo a la base de conocimiento y guía de procesos del **RUP** desde su desktop. La base de conocimiento unifica aún más al equipo identificando y asignando responsabilidades, artefactos y tareas de forma que cada miembro del equipo comprenda su contribución al proyecto. Unificando al equipo, se simplifica la comunicación, asegurando la asignación de recursos en forma eficiente, la entrega de los artefactos correctos, y el cumplimiento de los tiempos límite. (**Rational, 2007).**

Entrega del software operativo con confianza.

El **RUP** mantiene al equipo enfocado en producir incrementalmente software operativo a tiempo, con las características requeridas y con la calidad requerida. Las mejores prácticas probadas en la industria, contenidas en el **RUP**, incorporan las lecciones aprendidas de cientos de líderes de la industria y miles de proyectos. Ya no hay necesidad de re-inventar soluciones a desafíos de la ingeniería de software bien conocidos. Siguiendo el acercamiento al desarrollo iterativo del **RUP**, es posible entregar a tiempo y con confianza el software. **(Rational, 2007).**

Control de nuevas herramientas y tecnologías.

Características y Beneficios.

No existen dos proyectos de desarrollo de software que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo. **Rational Unified Process**, o **RUP**, es una plataforma flexible de procesos de desarrollo de software que ayuda proveyendo guías consistentes y personalizadas de procesos para todo el equipo de proyecto. **(RUP, 2003).**

1. **Las mejores prácticas más probadas de la industria** - Son las mejores prácticas de desarrollo adoptadas en cientos proyectos mundialmente y enseñadas como parte de la currículo en cientos de universidades, la metodología **RUP** se convirtió rápidamente en el estándar de facto para el proceso de desarrollo en la industria de software. (**Rational, 2007).**
2. **Proceso hecho práctico** - Diferente que otras metodologías comerciales, la plataforma **RUP** hace que el proceso sea práctico con bases de conocimiento y guías para ayudar en el despegue de la planificación del proyecto, integrar rápidamente a los miembros del equipo y poner en acción el proceso personalizado. (**RUP, 2003).**
3. **Se adapta a las necesidades de los proyectos** - Solo la plataforma **RUP** proporciona un framework de proceso configurable que permite seleccionar e implantar los componentes específicos de proceso necesarios para proporcionar un proceso consistente y customizado para cada equipo y proyecto. (**RUP, 2003).**

Una de las mejores prácticas centrales de **RUP** es la noción de desarrollar iterativamente. **Rational Unified Process** organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada workflow variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes releases ejecutables. (**Rational, 2007).**

Sistemas Operativos y Plataformas de Hardware Apropiadas

- HP-UX.
- Linux.
- Sun Solaris.
- Windows 2000.
- Windows NT.

- Windows XP.

1.6.2 XP (Extreme Programming)

Es la más relevante de todos los procesos ágiles de desarrollo de software formulada por Kent Beck. Los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Es utilizada para proyectos de corto plazo, equipo pequeño. Tiene como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

- Desarrollo iterativo e incremental: Pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas: Frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación por parejas: Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.

1.6.3 ¿Por qué RUP?

Se decidió utilizar RUP porque esta metodología mantiene al equipo enfocado en producir incrementalmente software operativo a tiempo, con las características requeridas y con la calidad requerida. El cual se divide en 4 fases y estas presentan iteraciones creando artefactos como objetivo fundamental de cada actividad, posee mucha documentación, muy organizativo basado en roles. Por todo lo antes dicho y teniendo como base lo expuesto anteriormente se puede concluir que este proceso de desarrollo es el idóneo para llevar a cabo la realización de este software.

1. 7 Herramientas de modelado visual

Para el modelado visual se utilizan herramientas que ayudan a la construcción de aplicaciones, estas sirven de apoyo para la realización de estos modelos, y por tanto, agilizan la elaboración de aplicaciones consistentes y duraderas.

1.7.1 Visual Paradigm

Visual Paradigm: Es una herramienta UML CASE visual que ayuda a construir aplicaciones rápidamente, mejor y económicamente.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. **(FREED, 2007).**Visual Paradigm es multiplataforma de ahí la necesidad imperiosa de modelar con una herramienta libre con el objetivo de la migración que la Facultad 8 pretende realizar ,utiliza UML como lenguaje de modelado, ayudando de una manera más rápida a la construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona una mejor interfaz gráfica de usuario y una mayor base de datos de esquema de apoyo, demostraciones interactivas de UML y proyectos UML, soporta generación de código y soporte de ingeniería inversa.

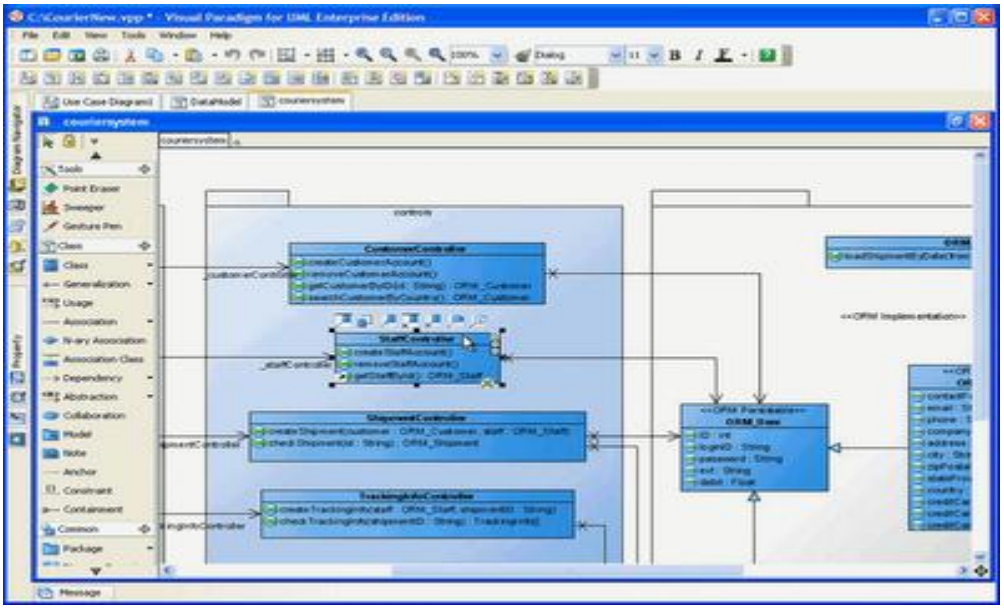


Figura 1.- Visual Paradigm for UML 6.4 Enterprise Edition

1.7.2 Rational Rose

Rational Rose: Es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. Rational Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a la monitorización del tiempo de desarrollo y al entendimiento del entorno de los sistemas.

1.7.3 ¿Por qué Visual Paradigm?

Se decide utilizar visual Paradigm porque presenta modelo y código que permanece sincronizado en todo el ciclo de desarrollo. Se tienen disponibilidad de múltiples versiones, para cada necesidad. También por su característica de ser multiplataforma lo que garantiza que se vea materializada la idea de la migración de la Facultad 8.

1.8 Lenguaje para la modelación.

Los lenguajes de modelado pretenden dar apoyo a la mayoría de los procesos de desarrollo de software, son desarrollados con el esfuerzo de simplificar y consolidar el gran número de métodos de desarrollo que han surgido.

1.8.1 UML (Unified Modeling Languaje)

Es un lenguaje grafico que le ayuda a especificar, visualizar y documentar modelos de sistemas de software, incluida su estructura y diseño, de manera que cumple con todos estos requisitos.

UML es un lenguaje más expresivo, claro y uniforme que otros anteriores que se utilizan para el diseño Orientado a Objetos, permite una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

Diagramas en UML

Diagramas de estructura: Incluyen el diagrama de clases, Diagrama de objetos, Diagrama de componentes, Diagrama de estructura compuesta, Diagrama de paquetes, y diagrama de despliegue.

Los diagramas de comportamiento: Incluyen el diagrama de casos de uso (algunos métodos utilizados por los requisitos durante la recolección), Diagrama de actividad, diagrama de la máquina y el Estado.

Los diagramas de interacción: Todos los derivados de la más general de comportamiento Diagrama, incluir el diagrama de secuencia, Diagrama de Comunicación, Calendario Diagrama, Interacción y Diagrama.

1.8.2 ¿Por qué UML?

Se selecciona UML para la modelación porque UML es una notación de entendimiento común para todo desarrollador, es vital para describir planos de software, definiendo un conjunto de reglas para representar modelos. Da la posibilidad de permitir documentar y especificar los elementos creados mediante un lenguaje común describiendo modelos. UML se ha convertido en lenguaje estándar para la realización de software.

1.9 Patrones

El Patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. **(Larman, 2004).**

1.9.1 Patrones de Diseño

Patrones de diseño: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software. Son una abstracción de una solución en un nivel alto solucionando problemas que existen en muchos niveles de abstracción.

1.9.1.1 Fachada

El uso de una clase de fachada para encapsular la complejidad de las interacciones entre los objetos de negocio y participantes en un flujo de trabajo. El patrón Fachada maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes.

Proporciona una interfaz unificada para un conjunto de interfaces en un subsistema, haciéndolo más fácil de usar. Reduce la complejidad y minimiza dependencias.

1.9.1.2 DAO

Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

El patrón DAO se encarga de ocultar totalmente los detalles de implementación de la fuente de datos a sus clientes.
El interface expuesto por DAO no cambia al cambiar la implementación de la fuente de datos subyacente.

1.9.1.3 GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa patrones generales de software para asignar responsabilidades.

Algunos Patrones GRASP

1. **Experto**

Experto es un patrón que se usa más que cualquier otro para asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la ¨ intuición ¨ de que los objetos hacen cosas relacionadas con la información que poseen. (**Larman, 2004).**

2. **Creador**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. (**Larman, 2004).**

3. **Alta Cohesión**

Es un principio de que se debe tener presente en todas las decisiones de diseño, es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. (**Larman, 2004).**

4. **Bajo acoplamiento**

El Bajo Acoplamiento estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultaos negativos propios de un alto acoplamiento, soporta el diseño de clases más independientes que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. (**Larman, 2004).**

5. **Controlador**

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Las operaciones del sistema deberían manejarse en la capa de dominio de los objetos y no en la interfaz, presentación o aplicación. (**Larman, 2004).**

1.9.2 Patrones de Caso de Uso

Los Patrones de Caso de Uso capturan técnicas para que el modelo sea mantenible, reusable y entendible, por lo que capturan mejores prácticas para modelar casos de uso. (**Cuesta, 2005).**

Beneficios

- Aumentar la productividad.
- Reutilizar elementos existentes (en este caso fragmentos de modelos).
- Evitar el retrabajo por errores.
- No invertir tiempo en resolver problemas ya resueltos.
- Aplicar la teoría al trabajo práctico.
- Habilitar las herramientas de soporte para modelar el desarrollo. **(Cuesta, 2005).**

Otros Patrones de Caso de Uso

- Reglas de Negocio.
- Login.
- Reportes y Explotación de Información.
- Inclusión y Extensión.
- Sistema en Capa.
- Múltiples Actores
- Jerarquía de Comportamiento
- Servicios Opcionales.

1.9.2.1 CRUD

El Patrón CRUD Completo consiste en un Caso de Uso para administrar la información (CRUD Información), permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y dar de baja. **(Cuesta, 2005).**

1.9.3 Patrones de Arquitectura

El uso de patrones arquitectónicos en un sistema informático en muchos casos conlleva a distribuir la lógica de aplicaciones en una computadora diferente, por otra parte y formando un ente importante ayudan a la reutilización de código y la posibilidad de representar la lógica en componentes aislados y también incluye la división de las responsabilidades.

1.9.3.1 Modelo Vista Controlador (MVC)

Conviene desacoplar los objetos del dominio (modelo) y las ventanas (vistas), a fin de brindar soporte a un mayor rehúso de los objetos de dominio y reducir al mínimo el impacto que los cambios de la interfaz tienen en ellos. **(EVA4, 2008).**

Definir las clases de dominio (modelo) para que no tengan acoplamiento n visibilidad directa respecto a las clases ventanas (vista) y para que los datos de la aplicación y de la funcionalidad se conserven en clases del dominio, no en la de ventana. **(EVA4, 2008).**

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulado sobre la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). (**EVA4, 2008).**

Vista: Maneja la visualización de la información. (**EVA4, 2008).**

Controlador: Controla el flujo entre la vista y el modelo(los datos). (**EVA4, 2008).**

1.10 Técnicas de Recopilación de Información

- Entrevistas
- Reuniones
- Cuestionarios
- Tormentas de Ideas

Las entrevistas y la reuniones constituyen la técnica principal de este trabajo para recopilar la información, aquellas fueron hechas con plena satisfacción del cliente y con el entendimiento común de ambos implicados para obtener las principales funcionalidades del sistema basado en los procesos de pruebas soportados.

1.11 Sistemas de Gestión de Bases de Datos (SGBD)

Los **Sistemas de gestión de base de datos** son un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario, facilitándole al usuario las herramientas que le permitan manipular o manejar de manera clara, sencilla y ordenada un conjunto de datos. Permiten la facilidad de manejo de grandes volúmenes de información alcanzando gran velocidad en breve tiempo, sin duplicaciones, aporta independencia del tratamiento de información y seguridad de la información.

1.11.1 MySQL

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Es la base de datos de código fuente abierto más usada del mundo. Presenta condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet. **(Peréz, 2007).**

Características

1. Utiliza múltiples tablas para almacenar y organizar la información.
2. Adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.
3. Velocidad.
4. Estabilidad.
5. Rapidez en ejecutar consultas.
6. Conectividad segura.
7. Multihilo.
8. Multiusuario.

1.11.2 PostgreSQL

PostgreSQL: Es un servidor de base de datos objeto relacional libre, es un motor de base de datos avanzado de código abierto. Es un sistema objeto-relacional, incluye características de la orientación a

objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Presenta las características esenciales de escalabilidad permitiendo gran cantidad de conexiones simultáneas y gran control de concurrencia al no permitir bloqueos innecesarios.

Este motor de base de datos soporta:

Querys complejos, incluyendo subselects

Integridad referencial (Foreign Keys)

Triggers

Vistas (Views)

Integridad Transaccional (ACID)

Control de versionado concurrente (MVCC)

Tiene licencia BSD

1.11.3 ¿Por qué PostgreSQL?

Se decide **PostgreSQL** porque presenta es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, permite la gestión de diferentes usuarios, permite la declaración de funciones propias incorporando funciones de diversa índole como el manejo de fecha y las operaciones con redes. PostqreSQL es rápido en ambos sentidos. Constituye una herramienta muy potente que permite grandes volúmenes de información.

1.12 Servidor web

Apache: Servidor de código abierto para plataformas *Unix* (BSD, GNU/*Linux*), *Windows* y otras, que implementa el protocolo HTTP y la noción de sitio virtual. Considerado el servidor número uno del mundo. (**KABIR, 2003).**

1. Apache no necesita grandes recursos en el ordenador para funcionar.
2. Multiplataforma: funciona en *Linux*, *Unix* y *Windows*.
3. Servidor altamente configurable de diseño modula.
4. Tecnología gratuita de código fuente abierta.
5. Trabaja con gran cantidad de código *script* como *Perl* y PHP.

6. Una vez que está instalado Apache, únicamente necesita unos 5 MB de espacio en el disco.

1.13 Lenguaje del lado del cliente

Existen diversas tecnologías del lado del cliente que permiten crear sitios según los estándares establecidos, los cuales ayudan a la presentación de los documentos y validaciones del contenido.

1.13.1 HTML

HTML (HyperText Markup Language) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, sonido.). Una de las características esenciales de este lenguaje es la universalidad, y significa que prácticamente cualquier ordenador, independientemente del sistema operativo, puede leer o interpretar una página web. Sin embargo, el aspecto de dichas páginas depende del tipo de ordenador, del monitor, la velocidad de la conexión de Internet y, por último del navegador. Aunque no todos los navegadores muestran una misma página web de la misma forma. (**EVA, 2007).**

1.13.2 CSS

CSS es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (**EVA1, 2007).**

1.13.3 JavaScript

JavaScript es el lenguaje de programación Web del lado del cliente más extendido. Con este lenguaje script se pueden generar páginas dinámicamente en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página. Fue creado por Netscape para su navegador 2.0 con la versión 1.0 de Javascript y ya marcha por la 1.5; está basado en objetos. **(EVA2, 2008).**

Por cuestiones de seguridad se ha impedido que JavaScript pueda leer y escribir a disco, ejecutar otros programas y conectarse a otra computadora o enviar correo, producto que pudiera introducirse virus de esta forma cuando un servidor envía una página a un cliente. De la misma forma en el navegador se puede configurar para no ejecutar código script. **(EVA2, 2008).**

1.14 Conclusiones

En este capítulo se abordó el estado del arte relacionado con las tendencias actuales, técnicas actuales , software utilizados, metodologías, y herramientas para crear software de calidad se puede definir que el proyecto de calidad de la Facultad 8 necesita un sistema que sea capaz de gestionar la calidad de los proyectos productivos por medio de los procesos de prueba por lo cual este sistema utilizará PHP como lenguaje de programación porque puede trabajar sobre Linux o cualquier servidor y como estrategia de la facultad de desarrollar aplicaciones sobre Linux esto constituye un paso de avance en la migración hacia el software libre, PostgreSQL como gestor de base de datos (BD), Apache como servidor Web, Rational Unified Process (RUP) como metodología de desarrollo donde se desarrollan proyectos grandes, RUP posee una aproximación iterativa ayudando a mitigar los riesgos en forma temprana, es adaptable a las necesidades del proyecto de calidad de la facultad porque permite seleccionar e implantar los componentes específicos de proceso necesarios para proporcionar un proceso consistente de pruebas. Visual paradigma-UML como herramienta para y Lenguaje Unificado de Modelado (UML) como lenguaje de modelado de sistemas de software que permitirá la creación de un conjunto de artefactos que facilitarán la implementación exitosa del sistema. Por estas razones expuestas se puede proponer el desarrollo de un sistema que, minimice el esfuerzo del grupo de trabajo en el proyecto de calidad ,centralice el proceso de desarrollo de pruebas por parte del equipo de prueba, proporcione una organización para que el administrador de prueba pueda detectar problemas que impidan las pruebas, realice un monitoreo del progreso y el resultado en cada ciclo de prueba, facilite la definición de los métodos de pruebas, proveer recursos correspondientes para las pruebas, permita la conducción de las pruebas y logre una retroalimentación de las pruebas.

Capítulo 2 Características del Sistema

2.1 Introducción

En el presente capítulo se conciben los objetivos estratégicos de la organización y procesos de negocio que los soportan, se hace una descripción de los procesos que serán objeto de automatización y de los sistemas automatizados que existen, así como la definición de la información que se maneja. Se obtiene además una descripción general de la propuesta de sistema y como debe funcionar. Se realiza un análisis comparativo con otras soluciones existentes, la descripción del modelo de negocio y la especificación de los requisitos del software.

2.2 Objeto de estudio

El objeto de estudio lo conforman los procesos de pruebas que tienen lugar en el proyecto de calidad de la Facultad 8, los cuales constituyen la piedra angular para la liberación de los productos informáticos y la aceptación de los clientes.

2.3 Problema y situación problémica

El Proyecto de Calidad de la Facultad 8 tiene la responsabilidad de guiar el proceso de gestión de la calidad en las fases de desarrollo de cada proyecto productivo para la liberación de productos terminados, formando la realización de pruebas un eslabón principal y un elemento crítico para la garantía de la calidad del software.

El proceso de pruebas en la Facultad 8 es muy importante para la elaboración y vida del software por lo que se debe hacer un esfuerzo especial en tratar de encontrar y corregir los errores antes de entregar el programa al cliente.

En el proyecto de calidad no se ha logrado una óptima gestión de las anomalías que se generan en cada iteración, se producen errores en cuanto al procesamiento de la información por la extensa documentación existente y la información necesaria para la realización de las pruebas no se encuentra centralizada. Existe dificultad con el llenado del control de versiones, lo que hace difícil identificar quienes cometen errores en la documentación que se entrega, impidiendo tomar medidas que impliquen rectificaciones a

tiempo. Todas estas dificultades se traducen en atrasos que repercuten negativamente en el cumplimiento de los cronogramas y liberación de dichos productos.

Por las razones antes expuestas, la dirección del Proyecto de Calidad identifica la necesidad de implementar un Sistema de Gestión de la Calidad para la automatización de los procesos de prueba.

2.3.1 Objetivos estratégicos de la organización y procesos de negocio que los soportan

El proyecto de Calidad de la Facultad 8 se propone garantizar la calidad de todos los proyectos productivos, presentando una herramienta automatizable que organice y guie el proceso de prueba por parte del equipo de prueba. Minimizar el esfuerzo del equipo de prueba, optimizar tiempo y costo de los proyectos que se liberan es la estrategia a seguir del proyecto de Calidad de la Facultad 8, debido a que actualmente el proceso de pruebas es imprescindible para que cada proyecto cuente con la calidad requerida por lo que el desarrollo correcto de este proceso se hace vital para el soporte de la organización.

2.3.2 Flujo actual de los procesos involucrados en el campo de acción

La Universidad de las Ciencias Informáticas tiene como aspiración convertirse en una gran industria de producción de software tanto en el ámbito nacional como internacional, para lograr este objetivo todas las facultades desarrollan proyectos productivos con varios perfiles para su informatización. También existe por cada facultad un proyecto de Calidad que es el encargado de garantizar la gestión de la calidad a todos los proyectos que se liberan en esta. En la actualidad la Facultad 8 presenta su proyecto de Calidad con la misión de contribuir a desarrollar proyectos con una alta calidad, por lo que se ha propuesto llevar a cabo un sistema informático con vistas a mejorar la gestión de la calidad de sus proyectos productivos.

La realización de las pruebas a los proyectos productivos es una tarea fundamental en el logro de las metas propuestas por parte del proyecto, una vez desarrollados los proyectos el equipo de desarrollo de la facultad le solicita realizar pruebas a Calidad UCI, donde Calidad UCI concilia la reunión pertinente en el cual es convocado el equipo de desarrollo y el proyecto de Calidad de la facultad donde se definen las pruebas a realizar a los proyectos productivos.

Una vez definida las pruebas a realizar es labor de Calidad UCI brindar capacitación al proyecto de Calidad, al capacitarse el proyecto de Calidad ya se encuentran en condiciones de realizar las pruebas a los proyectos. Calidad UCI es la encargada de revisar las que se encuentren en los proyectos productivos,

los cuales una vez revisados son enviados al equipo de desarrollo para que revise las anomalías o defectos encontrados en los proyectos.

El Equipo de desarrollo en cada etapa responde las anomalías y Calidad UCI recibe los Informes de Anomalía. Cuando Calidad UCI considera que se debe liberar el producto realiza un acta de liberación la cual es firmada por el Equipo de Desarrollo y Calidad UCI.

El proceso de pruebas tiene lugar cuando en la facultad el asesor de calidad hace una planificación de las pruebas y luego distribuye el trabajo. Los probadores son los encargados de ejecutar las pruebas utilizando herramientas de pruebas como casos de pruebas, listas de chequeo, una vez ejecutadas las pruebas se conforman los documentos de anomalías. Finalmente el coordinador centraliza y revisa los resultados obtenidos de la ejecución de las pruebas por parte del probador consultando las anomalías encontradas.

2.3.3 Análisis crítico

Actualmente el procedimiento general de pruebas en el proyecto de calidad de la Facultad 8 no se aplica completamente, pues solo se realizan pruebas de caja negra y a la documentación, el equipo de prueba al incluir las anomalías encontradas a estos elementos no presenta la organización que se necesita para hacer un trabajo eficiente por la diversidad de información que se encuentra descentralizada ,las pruebas a la documentación tienden a ser un poco tediosas y puede pasar que no sean detectados algunos defectos que presenta la documentación.

En cuanto a las pruebas a la aplicación se realizan pruebas exploratorias que son las encargadas de encontrar errores en la interfaz de usuario, es la forma manual la única vía hasta el momento para verificar la aceptabilidad de una prueba exploratoria pero se puede automatizar esta aceptación para tener un mayor control de las pruebas realizadas.

El desarrollo de las pruebas por parte del equipo de prueba para asegurar un satisfactorio proceso de pruebas y así garantizar la calidad de los proyectos la razón fundamental para que no exista retraso de tiempo para la liberación del producto y no se le entregue un producto de baja calidad al cliente y lograr la plena aceptación por su parte.

2.4 Objeto de automatización

En el proceso de prueba de un software es muy importante llevar a cabo el desarrollo de técnicas de prueba ,entre ellas pruebas de caja blanca las cuales se centran en la estructura de control del programa y las pruebas de caja negra diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa, es por ello la tentativa de automatizar el área de desarrollo de estas técnicas, poniendo en marcha la zona de organización de realización de las pruebas realizadas a los software especialmente la realización de pruebas de caja negra entre las que se encuentran las pruebas a los entregables ,los cuales pueden ser Documentación y Aplicación, dando posibilidad de la generación de reportes e informes una vez realizadas estas pruebas, tanto en la documentación como las pruebas exploratorias a las aplicaciones.

2.4.1 Descripción de los procesos

El proceso de pruebas constituye objeto de automatización en el sistema, la realización de las pruebas a los proyectos productivos forman parte de un ente fundamental para lograr el procesamiento de las diferentes pruebas realizadas a los software en la Facultad 8.La gestión de las anomalías y su posterior resolución, la centralización de los proyectos a probar y la generación de varios reportes muy importantes para la toma de decisiones y las mejoras continuas emiten la consolidación de un proceso automático para revertir los atrasos en los cronogramas y liberación de los productos.

2.4.2 Descripción de los sistemas automatizados existentes

En la Universidad de las Ciencias de la Informática (UCI) actualmente no existe un sistema automatizado que se encargue de gestionar la calidad total a los proyectos productivos. Se debe hacer alusión a la herramienta automatizada Trac, el cual es un software basado en la red de gestión de proyectos, proporciona una interfaz para sistemas de control de versiones, y un número de formas convenientes para permanecer en la parte superior de los acontecimientos y los cambios dentro de un proyecto.

Figura 2. - Herramienta Trac

Trac se distribuye bajo la licencia BSD modificada. El texto completo de la licencia se puede encontrar en línea, así como la copia en el archivo incluido en la distribución, este proyecto de código abierto permite el seguimiento de proyectos de desarrollo de software. Trac utiliza un enfoque minimalista basado en la web de software de gestión de proyectos. Su misión es ayudar a los desarrolladores a escribir software de gran mantenimiento fuera del camino, Ofrece un interfaz de Subversión (u otros sistemas de control de versiones), una presentación de informes de instalaciones.

Trac permite que las descripciones de marcas en cuestión, realiza una muestra de tiempo de todos los proyectos actuales. Crea eventos en orden haciendo que la adquisición de una visión general del proyecto

y el seguimiento de los progresos sean muy fácil. El plan de trabajo muestra el camino a seguir y la lista de las próximas etapas.

2.5 Información que se maneja

Existe gran cantidad de información en diferente documentos la cual posee detalles específicos manipulados y dependiendo del grado de importancia de la misma para su posterior procesamiento.

2.5.1 Documentos específicos manejados

- Proyecto a Revisar
- Plantillas de Diseño de Caso de Prueba
- Lista de Chequeo
- Plantillas de Anomalía
- Anomalía
- Informe de Anomalía
- Documentación
- Aplicación
- Interfaz Gráfica de Usuario
- Facilidades de Ayuda
- Priorización de Casos de Uso
- Control de Manuales
- Listado CU

2.6 Propuesta de sistema

El sistema contará con el módulo de pruebas el cual es el encargado de brindar la posibilidad al usuario de realizar los distintos tipos de prueba de caja negra ya sea a la aplicación o a la documentación. El módulo está compuesto por 9 submódulos, los cuales permiten a los usuarios realizar las actividades pertinentes en el sistema.

1. Módulo Pruebas: Es el área de trabajo donde se podrán gestionar las funcionalidades del sistema por parte de los usuarios, cuenta con diferentes bandejas para realizar funcionalidades sobre las aplicaciones y la documentación. En caso de las aplicaciones solo se realizan pruebas a la interfaz o

proyectos ejecutables y en caso de documentación se realizan pruebas a la documentación de los proyectos como son los Casos de Uso u otros elementos que soporten estas pruebas.

2. Sub-Módulo Prueba: Es el área que permite incluir, ver, modificar y eliminar una prueba por parte del jefe de prueba en el sistema.
3. Sub-Módulo Herramienta de Prueba: Es el área que permite gestionar las herramientas de prueba sirviendo de soporte para la realización de pruebas.
4. Sub-Módulo Anomalías: Es el área que permite al probador gestionar las anomalías a los distintos proyectos en prueba.
5. Sub-Módulo Solicitud: Es el área que permite gestionar solicitudes al jede de prueba y al líder de proyecto para llevar a cabo las pruebas a un proyecto.
6. Sub-Módulo Respuesta Anomalía: Es el área que permite al solicitante del equipo de desarrollo gestionar las respuestas de anomalías del proyecto probado
7. Sub-Módulo Reportes: Es el área que permite al jefe de prueba realizar los reportes referentes a los informes de anomalías y a los probadores y la generación de informes de anomalías.
8. Sub-Módulo Evaluación Actividades Turno: Es el área que permite obtener las evaluaciones de las actividades de los probadores en un determinado turno.
9. Sub-Módulo Distribución de Trabajo: Es el área que permite al jefe de prueba la distribución de trabajo de los probadores en un turno.

2.6.1 Descripción general de la propuesta de sistema

El sistema muestra gran avance en cuanto al proceso de prueba realizadas a los proyectos, cada nuevo proyecto que se le necesite gestionar su calidad es incluido en el área de trabajo del sistema. Una vez incluido es otorgado el permiso a los probadores para que se familiaricen con el proyecto a probar, y se le darán los artefactos a utilizar en ese proyecto durante el proceso de prueba, lo cual es labor del jefe de prueba proporcionales estos artefactos. Los usuarios que tienen privilegios altos como el jefe de prueba tendrán permiso para hacer modificaciones en el sistema. Los usuarios con privilegios medios como los probadores podrán interactuar con el sistema pero sin permiso de modificación en áreas restringidas.

2.6.2 Análisis comparativo

La Facultad 8 no posee una herramienta que sea capaz de gestionar la calidad de todos los proyectos productivos garantizándole la calidad requerida por parte del proceso de pruebas. La utilización del trac para la gestión de las anomalías se hace necesario y guía el trabajo del equipo de desarrollo ayuda al trabajo por parte del equipo de prueba pero no controla lo suficiente las actividades del personal para lograr un acercamiento al trabajo eficaz del equipo de prueba, se puede decir que esta herramienta realiza reportes pero no son suficientes porque los reportes tienen que ser más dirigidos al personal para que se pueda medir el progreso su progreso y así esto constituya un empuje parar la organización y realización del trabajo de cada miembro del equipo de prueba.

Uno de los principales problemas por lo que se realiza mal el trabajo o no se realiza, está dado por la poca motivación del personal para realizar sus tareas o la poca atención a las tareas a desarrollar. Con la realización de esta herramienta se piensa resolver estos problemas mediante una organización de las tareas concebidas en el proyecto proporcionando un área de desenvolvimiento del probador donde se sienta motivado y comprometido con las pruebas. El desarrollo del proceso de prueba realizado a cada proyecto por parte del probador debe ser eficiente a un 80% lo que significa que si cada probador emplea el 80% de su esfuerzo el proyecto cumple un 80 % o más de garantía de calidad.

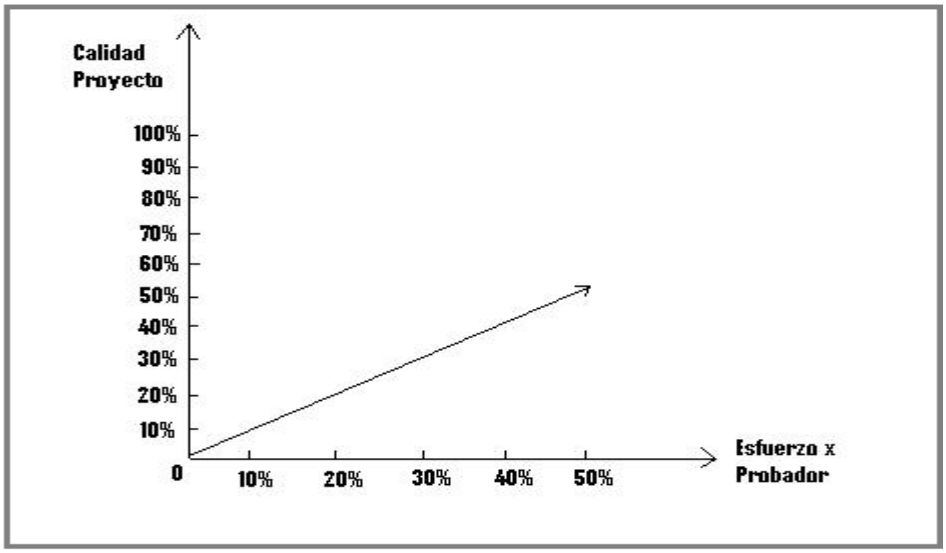


Figura 3.- Calidad de proyecto por esfuerzo de los probadores

2.7 Modelo de negocio

Un sistema, por pequeño que sea, generalmente es complicado. Por eso se necesita dividirlo en piezas si se pretende comprenderlo y gestionar su complejidad. Esas piezas se pueden representar a través de modelos que permitan abstraer sus características esenciales. Una técnica para la especificación de los requisitos más importantes del sistema, que da soporte al negocio, es el modelo del negocio, con lo cual se refuerza la idea de que sea el propio negocio lo que determine los requisitos. **(EVA3, 2008).**

La realización de pruebas es el proceso fundamental para garantizar la calidad de los proyectos ,la información descentralizada par realizar las pruebas , el no llenado correctamente del control de versiones ,el desempeño no óptimo de la gestión de las anomalías, la extensa documentación y la falta de motivación del personal para probar quedaría atrás con un sistema que garantice la centralización de toda la información de los proyectos a probar, proporcionando la correcta manipulación con el control de versiones, la existencia de poca documentación en la estación de trabajo, dando al traste el aumento de la motivación por parte del personal por contar con un ambiente más agradable para la realización de las pruebas pretendiendo consolidar al máximo el desempeño de optar por un software libre de errores a un alto grado.

2.7.1 Reglas del Negocio a considerar

Reglas de Estructura

- **LCH**: (Lista de chequeo).Artefacto que se utiliza para la realización de las pruebas.
- **Informe_Fin_Iter**: Informe de fin de iteración creado por el jefe de prueba al culminar una iteración de prueba.
- Interesados: Puede ser el Jefe de Polo, el Líder de Proyecto o el Vicedecano de Producción.
- **Plan_Prueba_Proyecto**: Plan de prueba creado por el jefe de prueba y los interesados para la realización de las pruebas.
- Una herramienta de prueba pude ser una Lista de Chequeo o una Plantilla de Diseño de Caso de Prueba.
- Un entregable puede ser un documento o la aplicación.

Reglas de Restricción:

- Un probador solo puede gestionar las anomalías que él tiene asociado.
- El jefe de prueba puede hacer cualquier operación.
- El rol Jefe de Prueba solo puede ser asignado a un usuario del sistema.
- La versión de una prueba incluida por un probador no puede ser una versión fuera de rango en la iteración perteneciente.

Reglas de Derivación:

- Cuando un probador no realiza su trabajo completo se considera que este incumplió con el mismo.
- Cuando un probador deja trabajo por hacer se considera que el trabajo es trabajo pendiente.
- Cuando un probador realiza su trabajo completo se considera que cumplió.
- La cantidad total de las anomalías detectadas por un probador es igual a la suma de las anomalías detectadas en todos los proyectos que ha probado.

Reglas de Acción:

- Un Solicitante del equipo de desarrollo realiza una solicitud de prueba al Proyecto de Calidad, el Proyecto de Calidad verifica si este proyecto está en prueba y de no estarlo lo admite.
- A un proyecto que se encuentra en pruebas pasa por varias iteraciones de esa misma prueba.
- A un proyecto que no ha pasado por pruebas se le inicia el proceso de pruebas.
- Si un probador no asiste a su turno de trabajo en hora y fecha se procede a registrar su inasistencia de trabajo.

2.7.2 Actores del Negocio

Tabla 1.- Actor del negocio Lider_Proyecto

Actores del negocio	**Justificación**

Lider_Proyecto	Encargado de proveer la información necesaria por parte del proyecto y mediante el Desarrollador, para apoyar las pruebas al producto, discute junto con el jefe de pruebas las respuestas a las anomalías detectadas, se mantiene informado a través de los reportes, del estado de las pruebas, recibe la notificación si el producto está liberado o no.

Tabla 2.- Actor del negocio Vicedecano_Produccion

Actores del negocio	Justificación
Vicedecano_Produccion	El Vicedecano participa en la conformación del cronograma, media entre los interesados y el Jefe de Pruebas, evaluando y sugiriendo en dependencia de las necesidades inmediatas de la facultad. Garantiza la coordinación de las actividades y las condiciones en la que esta se desarrolla, se mantiene informado sobre el estado de las pruebas a través de los reportes enviados por el Jefe de Pruebas.

Tabla 3.- Actor del negocio Solicitante

Actores del negocio	Justificación
Solicitante	Es el responsable de hacer solicitudes al proyecto de calidad para la realización de pruebas. Puede ser el Lider_Proyecto o el Vicedecano_Produccion.

2.7.3 Trabajadores del Negocio

Tabla 4.- Trabajador del negocio Representante_ED

Trabajadores del negocio	Justificación
Representante_ED	Representa y responde por los intereses del Equipo de Desarrollo mediando entre este y el Equipo de

	Pruebas. Imparte capacitación previa a las pruebas, sobre el producto. Permanece durante las pruebas junto al Equipo de Pruebas, ratificando las detectadas, explicando elementos de difícil comprensión del Producto y comprobando el avance de las pruebas. Encargado de recoger las anomalías detectadas y de entregar al Jefe de Pruebas las respuestas a esas anomalías. Participa en la conciliación antes de cada iteración.

Tabla 5.- Trabajador del negocio Jefe_Prueba

Trabajadores del negocio	**Justificación**
Jefe_Prueba	Define el Cronograma de pruebas y lo actualiza mensualmente con la información que le brindan los Jefes de Polo. Elabora el plan de prueba general y un plan de prueba por proyecto. Realiza los reportes diarios de la jornada de trabajo y los envía a los interesados. Además realiza el informe de fin de iteración y finalmente emite un informe de liberación del proyecto a los interesados sobre la liberación o no, del producto probado. Distribuye y asigna el trabajo a los Probadores. Revisa las plantillas de Anomalías. Envía las anomalías detectadas. Discute con los interesados las respuestas a esas anomalías.

Tabla 6.- Trabajador del negocio Probador

Trabajadores del negocio	**Justificación**
Probador	Es el responsable de detectar y registrar las anomalías a los proyectos que se prueben. Además, los probadores envían las Plantillas de Anomalías al Jefe de Pruebas. Utiliza las Plantillas_DCP o las Listas de Chequeo para realizar las pruebas.

Tabla 7.- Trabajador del negocio Diseñador_CPLCH

Trabajadores del negocio	**Justificación**
Diseñador_CPLCH	Es el responsable del proyecto de calidad realizar los diseños de caso de prueba y las listas de chequeos de los proyectos que se prueben. Puede detectar y registrar en las pruebas. Además, los diseñadores, envían las plantillas de al Jefe de Prueba. Utiliza las Plantillas de Diseño de Caso de Prueba o las Listas de Chequeo para realizar las pruebas.

Tabla 8. - Trabajador del negocio Especialista

Trabajadores del negocio	**Justificación**
Especialista	Es el responsable de orientar al equipo de pruebas sobre aspectos que no estén claros para la realización las pruebas utilizando herramientas de prueba.

Tabla 9. - Trabajador del negocio Representante_EP

Trabajadores del negocio	**Justificación**
Representante_EP	Es el responsable del equipo de prueba de orientar sobre odas las herramientas a utilizar para la realización de las pruebas.

Tabla 10.- Trabajador del negocio Jefe_Polo

Trabajadores del negocio	**Justificación**
Jefe_Polo	El jefe de Polo provee al Jefe de Pruebas de la información necesaria para definir el Cronograma de Pruebas (garantizando que ésta, sea lo más completa posible) y envía una actualización de la misma mensualmente al Jefe de Pruebas.

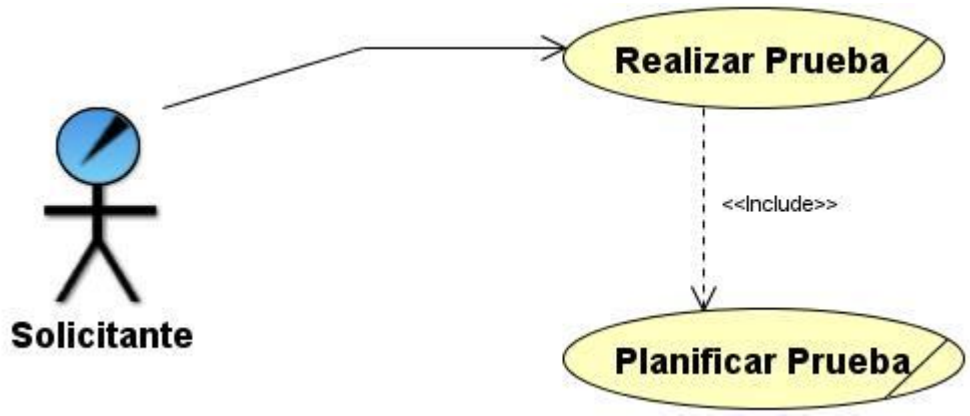


Figura 4.- Diagrama de Caso de Uso del Negocio

2.7.4 Listado de casos de uso del negocio.

Tabla 11. – Caso de Uso del Negocio Planificar Prueba

CU-1	**Planificar Prueba (incluido)**
Actores	Lider_Proyecto
Propósito	Planificar las pruebas que se realizan a los proyectos.
Resumen	El caso de uso se inicia cuando se solicita realizarle pruebas a un producto.
Flujo Normal de los Eventos	
Acción del Actor	**Respuesta del Negocio**
	1. El Jefe_Prueba y Jefe_Polo establecen el Cronograma de Pruebas.
	2. El Jefe_Prueba redacta y envía el Plan de Pruebas.
3. El Líder_Proyecto garantiza los entregables y la participación de un representante del equipo de desarrollo para la preparación del	

Equipo de Pruebas.	
	4. El Jefe_Prueba comunica al Equipo de Pruebas el comienzo de la preparación.
	5. El Representante_ED imparte una preparación del negocio sobre el producto que va a entrar en pruebas.
	6. Los Diseñadores_CPLCH inician la elaboración de las herramientas necesarias.
	7. El Jefe_Prueba solicita la presencia de un Especialista_DC para aclarar dudas en cuanto a la elaboración de las herramientas.
	8. El Especialista_DC orienta al equipo de pruebas sobre aspectos que no estén claros para la realización de esas actividades.
	9. Los Diseñadores_CPLCH realizan los diseños de casos de pruebas y/o las listas de chequeo culminando así el Caso de Uso.
Flujo Alterno de los Eventos	
6. a Los diseñadoresCPLCH no necesitan elaborar herramientas.	
Acción del Actor	**Respuesta del Negocio**
	6. a.1 Los Diseñadores_CPLCH no necesitan elaborar herramientas nuevas para realizar las pruebas, se revisan las herramientas utilizadas anteriormente que se ajustan a las características del producto a probar, culminando así el Caso de Uso.

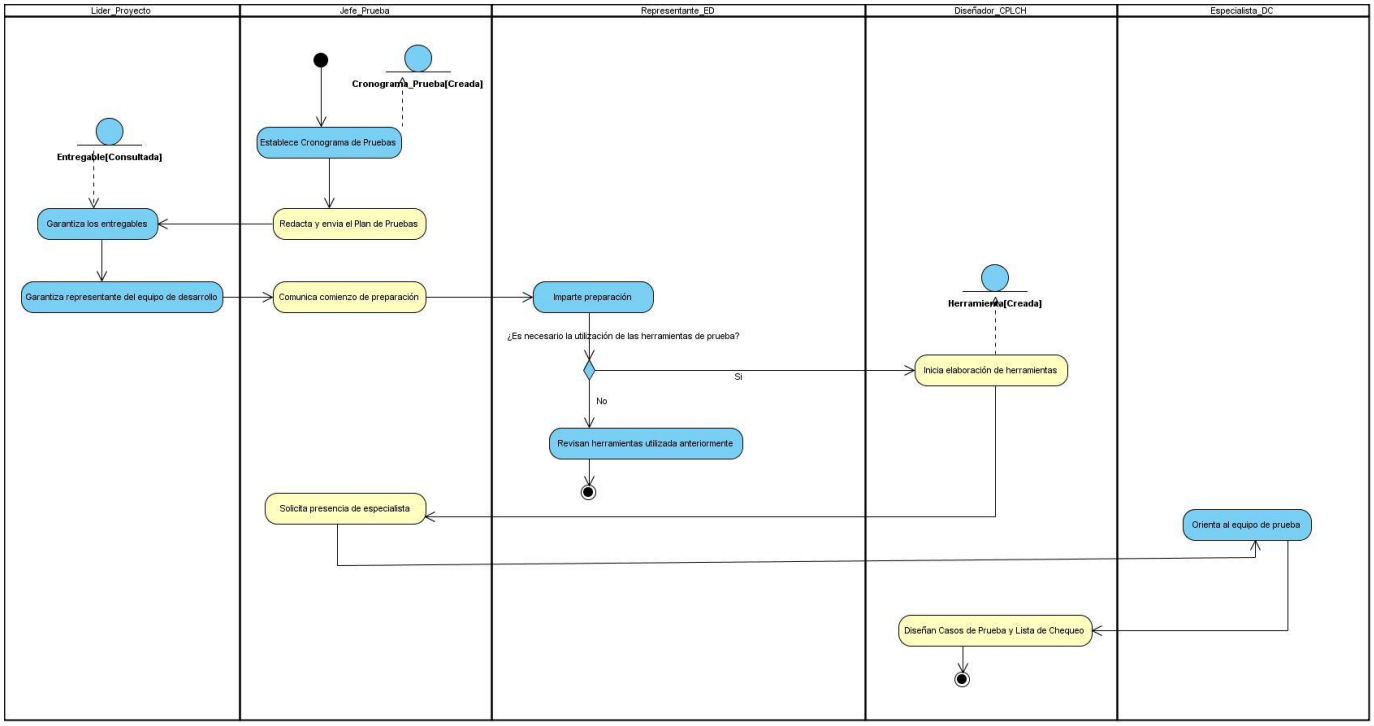


Figura 5.- Diagrama de actividades del caso de uso del negocio Planificar Prueba

Tabla 12.- Caso de Uso del Negocio Planificar Prueba Realizar Prueba

Caso de Uso	**Realizar Prueba**
Actores	Lider_Proyecto
Propósito	Realizar una prueba al proyecto que está en prueba.
Resumen	El caso de uso se inicia cuando se solicita realizarle pruebas a un producto.
Flujo Normal de los Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El Lider_Proyecto realiza una solicitud de pruebas para la liberación de sus productos al Jefe_Polo.	

	2. El Jefe_Polo agrupa y revisa las solicitudes recibidas para la liberación de los productos de su Polo enviándolas al Jefe_Prueba.
	3. El Jefe_Prueba planifica las pruebas. **(Ver CU Incluido Planificar Prueba).**
	4. El Jefe_Prueba distribuye el trabajo a cada Probador.
	5. El Probador realiza una Prueba Exploratoria en la que registra las anomalías detectadas en la Plantilla de Anomalías la que envía al Jefe de Pruebas al culminar con el trabajo orientado.
	6. El Jefe_Prueba revisa cada Plantilla de Anomalías recibida verificando la calidad del trabajo.
	7. El Jefe_Prueba verifica la completitud del trabajo correspondiente a la jornada laboral sin asignar nueva distribución de trabajo.
	8. El Jefe_Prueba envía las Plantillas de Anomalías y un Reporte al finalizar la jornada laboral.
9. El Lider_Proyecto recibe las Plantillas de Anomalías y el Reporte.	
10. El Líder_Proyecto envía respuesta de las Anomalías detectadas.	
	11. El Jefe_Prueba recibe las Plantillas de Anomalías respondidas para su revisión.
	12. El Jefe_Prueba convoca al Lider_Proyecto a una reunión de

	conciliación de estado de anomalías antes de iniciar cada iteración, logrando un acuerdo en el trabajo.
	13. El Jefe_Prueba inicia la primera de las iteraciones oficiales al producto siguiendo el mismo procedimiento que para la realización de la Prueba exploratoria. Regresa al paso 4 del Flujo Normal de los Eventos.
	14. El Equipo de Pruebas ha realizado más de 3 iteraciones de pruebas y no ha detectado más anomalías.
	15. El Jefe_Prueba notifica al Lider_Proyecto que el producto puede ser liberado.
16. El Líder_Proyecto recibe la notificación y el CUN termina.	

Flujo Alterno de los Eventos	
6. a El Jefe_Prueba detecta errores en la Plantilla de Anomalías.	
Acción del Actor	**Respuesta del Negocio**
	6. a.1 El Jefe_Prueba localiza errores en la Plantilla de Anomalías reenviándole la misma al probador que ejecutó las pruebas.
	6. a.2 Regresa al paso del Flujo Normal de los Eventos.

Flujo Alterno de los Eventos	
7. a El Jefe_Prueba verifica que no ha concluido el trabajo a asignar correspondiente a la jornada laboral.	
Acción del Actor	**Respuesta del Negocio**
	7. a.1 El Jefe_Prueba distribuye nueva ronda de trabajo a los probadores hasta concluir con lo correspondiente a la jornada de trabajo.

	7.a.2 Regresa al paso del Flujo Normal de los Eventos
Flujo Alterno de los Eventos	
12. a El Jefe_Prueba no establece acuerdo de trabajo con el Lider_Proyecto.	
Acción del Actor	**Respuesta del Negocio**
	12. a.1 El Jefe_Prueba y Lider_Proyecto no llegan a un acuerdo en la reunión de conciliación y se abortan las pruebas.
	12. a.2 Culmina el Caso de Uso.

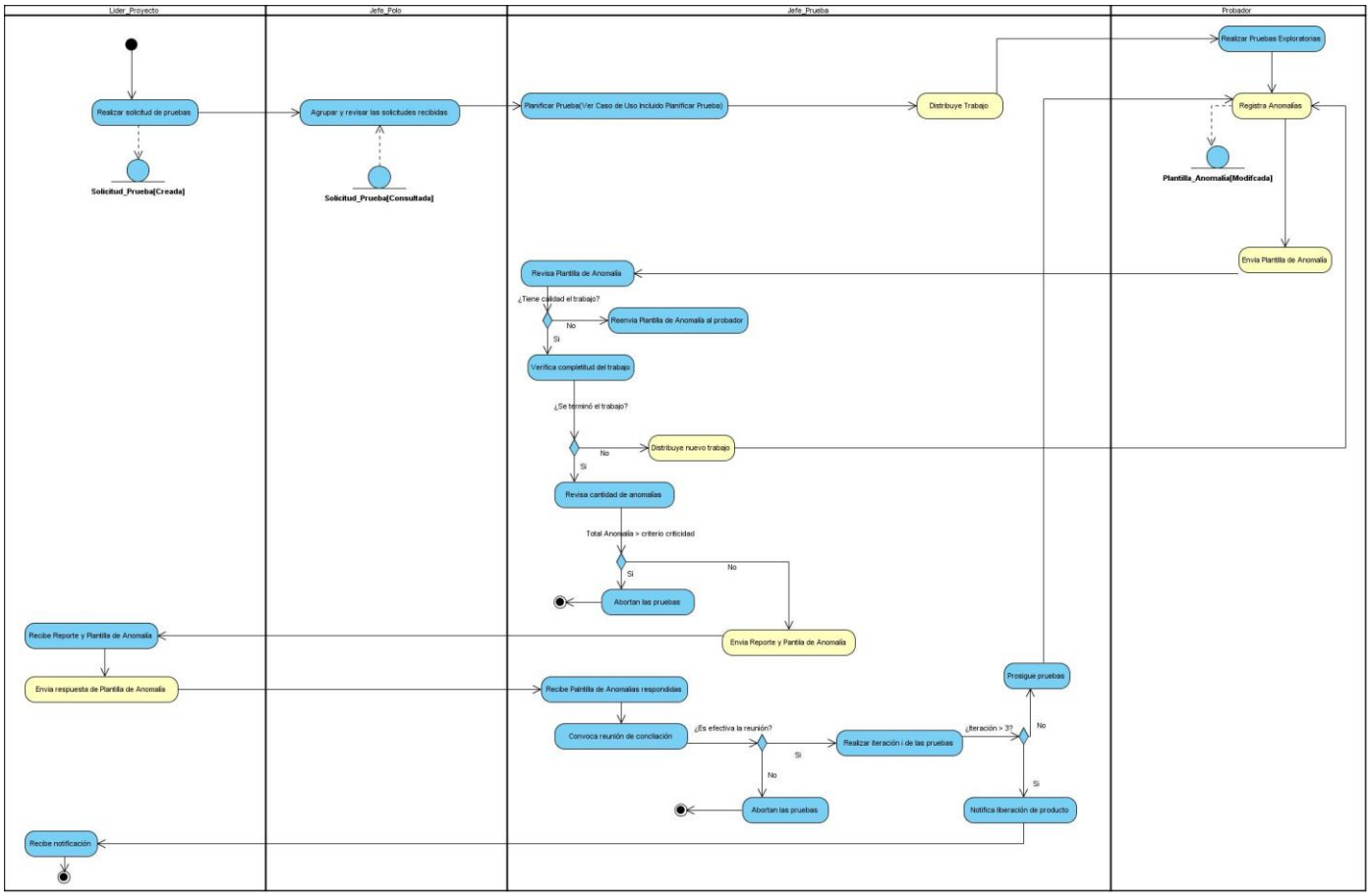


Figura 6.- Diagrama de actividades del caso de uso del negocio Realizar Prueba

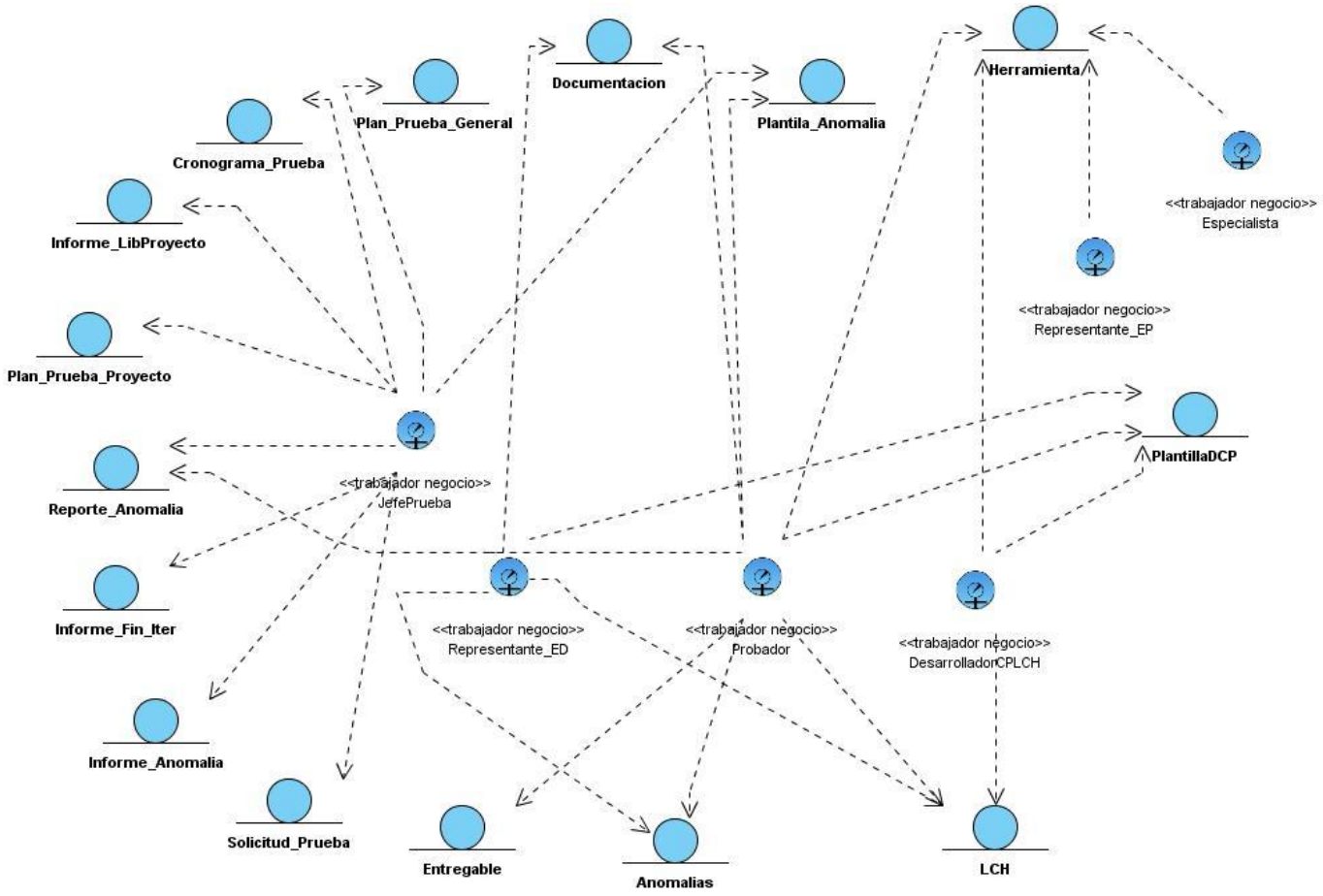


Figura 7.- Diagrama de Clases del Modelo de Datos

2.8 Especificación de los requisitos de software.

Los requisitos de software deben ser especificados para su entendimiento, la captura de requisitos es una de las actividades fundamentales que se desarrollan en el Flujo de Requerimientos durante la fase de inicio del desarrollo de un software, existen los requisitos funcionales y los no funcionales.

2.8.1 Requerimientos Funcionales

Son capacidades o condiciones que el sistema debe cumplir.

Características del Sistema

En la realización de los casos de uso del negocio, se obtienen las actividades que serán objeto de automatización. Estas actividades no son exactamente los requerimientos funcionales, pero sí son el punto de partida para identificar qué debe hacer el sistema. **(EVA5, 2008).**

Con el propósito de satisfacer las necesidades del grupo de calidad de la facultad 8 y las funcionalidades que en ella se ejecutan se proponen los siguientes requisitos funcionales:

Requisitos que debe cumplir el sistema:

RF1- Gestionar Prueba

RF1.1- Incluir Prueba.

RF1.2- Ver Prueba.

RF1.3- Modificar Prueba.

RF1.3- Eliminar Prueba.

RF2- Gestionar Lista de Chequeo

RF2.1- Incluir Lista Chequeo.

RF2.2- Ver Lista Chequeo.

RF2.3-Modificar Lista Chequeo.

RF2.4-Eliminar Lista Chequeo.

RF3- Gestionar Diseño de Caso de Prueba

RF3.1-Incluir Diseño de Caso de Prueba.

RF3.2-Ver Diseño de Caso de Prueba.

RF3.3-Modificar Diseño de Caso de Prueba.

RF3.4-Eliminar Diseño de Caso de Prueba.

RF4 - Gestionar Anomalía

RF4.1-Incluir Anomalía.

RF4.2-Ver Anomalía.

RF4.3-Modicar Anomalía.

RF4.4-Eliminar Anomalía.

RF5-Gestionar Solicitud

RF5.1- Incluir Solicitud.

RF5.2- Ver Solicitud.

RF5.3- Modificar Solicitud.

RF5.3- Eliminar Solicitud.

RF6-Getionar Respuesta Anomalía

RF5.1-Incluir Respuesta Anomalía.

RF5.2-Ver Respuesta Anomalía.

RF5.3-Modificar Respuesta Anomalía.

RF5.4-Eliminar Respuesta Anomalía.

RF7-Consultar Reporte

RF7.1 - Ver Reporte de Probador por Proyecto.

RF7.2 - Ver Reporte de Probador en Proyecto.

RF7.3- Generar Informe Anomalía por Rango de Fecha.

RF8-Evaluar Actividades Turno

RF9-Distribuir Trabajo

RF9.1 –Ver Trabajo Pendiente.

RF9.2 –Asignar Trabajo Pendiente.

2.8.2 Requerimientos no Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. (**EVA5, 2008).**

Con el propósito de satisfacer las necesidades del grupo de calidad de la facultad 8 se propone como requisitos no funcionales los siguientes:

Apariencia o Interfaz Externa:

- La herramienta debe ser sencilla, fácil de emplear y muy amigable para el usuario sin saturación de colores ni imágenes.
- El sistema debe poseer una interfaz Web amigable y sencilla, fácil para la interacción del usuario. Con colores en su interfaz que distingan a la facultad, sin saturación de colores ni imágenes.

Usabilidad:

- La herramienta está orientada al responsable de ocupar el rol de jefe de prueba en la facultad 8, a los diseñadores de Caso de Prueba y Listas de Chequeo, al solicitante del equipo de desarrollo y a los probadores.
- La herramienta será desplegada en un entorno web.
- Constituye una potente herramienta para realizar las pruebas.

Rendimiento:

- Operaciones normales en dos hilos
- Respuestas rápidas en segundos al usuario.
- Debe estar disponible las 24 horas del día.

Soporte:

- Documentación necesaria para el uso o desarrollo de las funcionalidades de la herramienta.

Portabilidad:

- El sistema debe ejecutarse en Windows XP 2000 u otro superior y en cualquier distribución GNU/Linux, es decir el sistema debe ser multiplataforma.

Seguridad:

- El sistema debe proteger la integridad de la información que se maneja.
- El sistema no debe presentar fallos, y en caso contrario minimizar las pérdidas de información.
- Los servidores deben permanecer con un cuidado extremo en los laboratorios de producción.

- Contemplar la seguridad física del lugar donde se usa la aplicación.
- Se permitirá la autenticación, la cual será una contraseña de acceso para los usuarios autorizados por el administrador. El sistema informático debe garantizar que la información sea vista únicamente por quien tenga permiso para esto. Él sistema debe permitir solo el acceso a cada uno de sus servicios al personal requerido para esto.
- El sistema informará quien es el usuario correspondiente con la sección que esté activa.

Políticos-culturales:

- No debe contener palabras en otros idiomas.
- Se deben poner las palabras lo más acorde a su entendimiento.

Legales:

- Es un deber proteger la información por parte de las personas que tienen derecho de administración u otros que pongan en peligro la integridad y seguridad del sistema.

Confiabilidad:

- Está prohibido la diseminación pública de la información por cualquier usuario a cualquier nivel.

Confiabilidad:

- Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

Disponibilidad:

- El sistema debe permitirle al usuario conectarse desde cualquier PC de la red.

Soporte:

- La aplicación tendrá un periodo de tiempo en prueba para ver su funcionamiento y que cumpla con lo requerido dándole mantenimiento para luego brindar servicio de instalación.
- Documentación necesaria para el uso o desarrollo de las funcionalidades de la herramienta.

Integridad:

- La información manejada por el sistema será objeto de cuidadosa protección

Interfaz:

- Las interfaces deben ser lo más entendible posible.
- Todas las interfaces tienen que tener una forma de cerrarse o retornar al paso anterior.
- Ninguna interfaz puede ir cargada de muchos colores evitando la desconcentración del usuario.

Software:

- Servidor Web Apache v1.x o superior.
- SGBD: PostgreSQL 8.2
- Navegador Internet Explorer v4.0 o superior.
- Tener como sistema Operativo Windows 98 o superior, Linux o Unix.

Requerimientos de Hardware:

- Se requiere de un servidor de 256 MB de RAM como espacio mínimo y 1GigaByte de espacio libre en disco duro.
- Las computadoras para el proceso de pruebas deben estar conectadas a una red y tener al menos 128 MB de RAM.
- Impresora EPSON.
- Tarjeta de red.

Ayuda y documentación en línea:

- La aplicación tiene que llevar un manual de ayuda.

2.9 Definición de los casos de uso

Los casos de uso representan las secuencias de acciones que es capaz de ejecutar un sistema produciendo un resultado observable por un actor.

2.9.1 Representación de los actores

Tabla 13.- Actor del Sistema Probador

Actores	Justificación
Probador	Es el responsable de detectar y registrar las anomalías de los proyectos que están probándose, tanto a las aplicaciones como a la documentación.

Tabla 14.- Actor del Sistema Diseñador_CPLCH

Actores	Justificación
Diseñador_CPLCH	Es el responsable de realizar los diseños de caso de prueba y las listas de chequeo, donde debe velar por la integridad de los mismos para obtener diseños consistentes.

Tabla 15.- Actor del Sistema Jefe_Prueba

Actores	Justificación
Jefe_Prueba	Es el responsable de realizar las pruebas al producto, revisa las anomalías, realiza un informe de anomalías y elabora el resumen del día. Evalúa y documenta el resultado de las pruebas realizadas al software. Debe poseer conocimientos básicos sobre RUP, UML y pruebas de software.

Tabla 16.- Actor del Sistema Lider_Proyecto

Actores	Justificación
Lider_Proyecto	Es el responsable del equipo de desarrollo de realizar las solicitudes al proyecto de calidad

	para la creación de las pruebas de software por parte del Proyecto de Calidad de la Facultad 8.

Tabla 17.- Actor del Sistema Solicitante_ED

Actores	**Justificación**
Solicitante_ED	Es el responsable del equipo de desarrollo de responder las anomalías para ser enviadas al equipo de prueba.

2.9.2 Listado de Casos de Uso

La representación de un listado de casos de uso para exponer los casos de uso del sistema se hace referencia posteriormente.

Tabla 18.- Caso de Uso Gestionar Prueba

CU-1	Gestionar Prueba
Actor	Jefe_Prueba
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Prueba. El actor puede crear, ver, modificar y eliminar una Prueba. En caso de que seleccione la opción de crear una prueba, el sistema da la posibilidad de introducir los datos necesarios para crear una Prueba. Si el actor elige la opción de ver una prueba el sistema muestra el contenido de la prueba en cuestión. Si el actor elige la opción de modificar una Prueba., el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar un aprueba el sistema elimina la prueba en cuestión.
Referencia	RF1

Tabla 19.- Caso de Uso Gestionar Lista de Chequeo

CU-2	Gestionar Lista de Chequeo
Actor	Diseñador_CPLCH

Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Lista de Chequeo. El actor puede incluir, ver, modificar y eliminar una Lista de Chequeo. En caso de que seleccione la opción de incluir una Lista de Chequeo, el sistema da la posibilidad de introducir los datos necesarios para incluir una Lista de. Si el actor elige la opción de ver una Lista de Chequeo, el sistema muestra el contenido de la Lista de Chequeo en cuestión. Si el actor elige la opción de modificar una Lista de Chequeo. El sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Lista de Chequeo el sistema elimina la Lista de Chequeo en cuestión.
Referencia	RF2

Tabla 20.- Caso de Uso Gestionar Diseño de Caso de Prueba

CU-3	Gestionar Diseño de Caso de Prueba
Actor	Diseñador_CPLCH
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un Diseño de Caso de Prueba. El actor puede incluir, ver, modificar y eliminar un Diseño de Caso de Prueba. En caso de que seleccione la opción de incluir una Lista de Chequeo, el sistema da la posibilidad de introducir los datos necesarios para incluir una Lista de Chequeo. Si el actor elige la opción de ver una Lista de Chequeo, el sistema muestra el contenido de la Lista de Chequeo en cuestión. Si el actor elige la opción de modificar una Lista de Chequeo, el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Lista de Chequeo, el sistema elimina la Lista de Chequeo en cuestión.
Referencia	RF3

Tabla 21.- Caso de Uso Gestionar Anomalía

CU-4	Gestionar Anomalía
Actor	Jefe_Prueba,Probador
Descripción	El caso de uso inicia cuando el probador selecciona la opción que le permite realizar una acción sobre una Anomalía. El probador puede incluir, ver, modificar y eliminar una Anomalía. En caso de de que seleccione la opción de incluir una Anomalía, el sistema da la posibilidad de introducir los datos necesarios para incluir una Anomalía. Si el actor elige la opción de ver Anomalía el sistema muestra el contenido de dicha Anomalía. Si el actor selecciona la opción de modificar Anomalía. Si el actor selecciona la opción de eliminar una Anomalía el sistema

	elimina la anomalía en cuestión.
Referencia	RF4

Tabla 22.- Caso de Uso Gestionar Solicitud

CU-5	Gestionar Solicitud
Actor	Jefe_Prueba,Lider_Proyecto
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Solicitud. El actor puede incluir, ver, modificar y eliminar una Solicitud. En caso de que seleccione la opción de incluir una Solicitud, el sistema da la posibilidad de introducir los datos necesarios para incluir una Solicitud. Si el actor elige la opción de ver una Solicitud el sistema muestra el contenido de la Solicitud en cuestión. Si el actor elige la opción de modificar una Solicitud., el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Solicitud el sistema elimina la Solicitud en cuestión.
Referencia	RF5

Tabla 23.- Caso de Uso Gestionar Repuesta Anomalía

CU-6	Gestionar Respuesta Anomalía
Actor	Solicitante_ED
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Respuesta de anomalía. El actor puede incluir, ver, modificar y eliminar una Respuesta de Anomalía. En caso de que seleccione la opción de crear una Respuesta de Anomalía, el sistema da la posibilidad de introducir los datos necesarios para crear una Respuesta de Anomalía. Si el actor elige la opción de ver una Respuesta de Anomalía el sistema muestra el contenido de la Respuesta de Anomalía en cuestión. Si el actor elige la opción de modificar una Respuesta de Anomalía., el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Respuesta de Anomalía el sistema elimina la Solicitud en cuestión.
Referencia	RF6

Tabla 24.- Caso de Uso Consultar Reporte

CU-7	Consultar Reporte
Actor	Jefe_Prueba
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un Reporte. El actor puede ver reporte de anomalía por turno, ver reporte probador, generar informe anomalía por fecha. En caso de que seleccione la opción de ver reporte de anomalía por turno, el sistema da la posibilidad de hacer el reporte correspondiente escogiendo el turno asociado. Si el actor elige la opción de ver reporte probador, el sistema muestra el reporte correspondiente escogiendo el probador asociado. Si el actor elige la opción de generar informe anomalía por fecha, el sistema muestra el informe de anomalía correspondiente asociado a la fecha escogida.
Referencia	RF7

Tabla 25.- Caso de Uso Evaluar Actividades por Turno

CU-8	Evaluar Actividades por Turno
Actor	Jefe_Prueba
Descripción	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre evaluar actividades por turno. El actor puede evaluar las actividades por turno de un probador escogiendo el turno y el probador, el sistema da la genera una evaluación dependiendo de las actividades del probador en el turno.
Referencia	RF8

Tabla 26.- Caso de Uso Distribuir Trabajo

CU-9	Distribuir Trabajo
Actor	Jefe_Prueba
Descripción	El caso de uso inicia cuando el Jefe_Prueba accede a la sesión de trabajo para la distribución de trabajo, selecciona la opción de Distribuir trabajo y realiza la distribución a los probadores existentes en el turno de trabajo.
Referencia	RF9

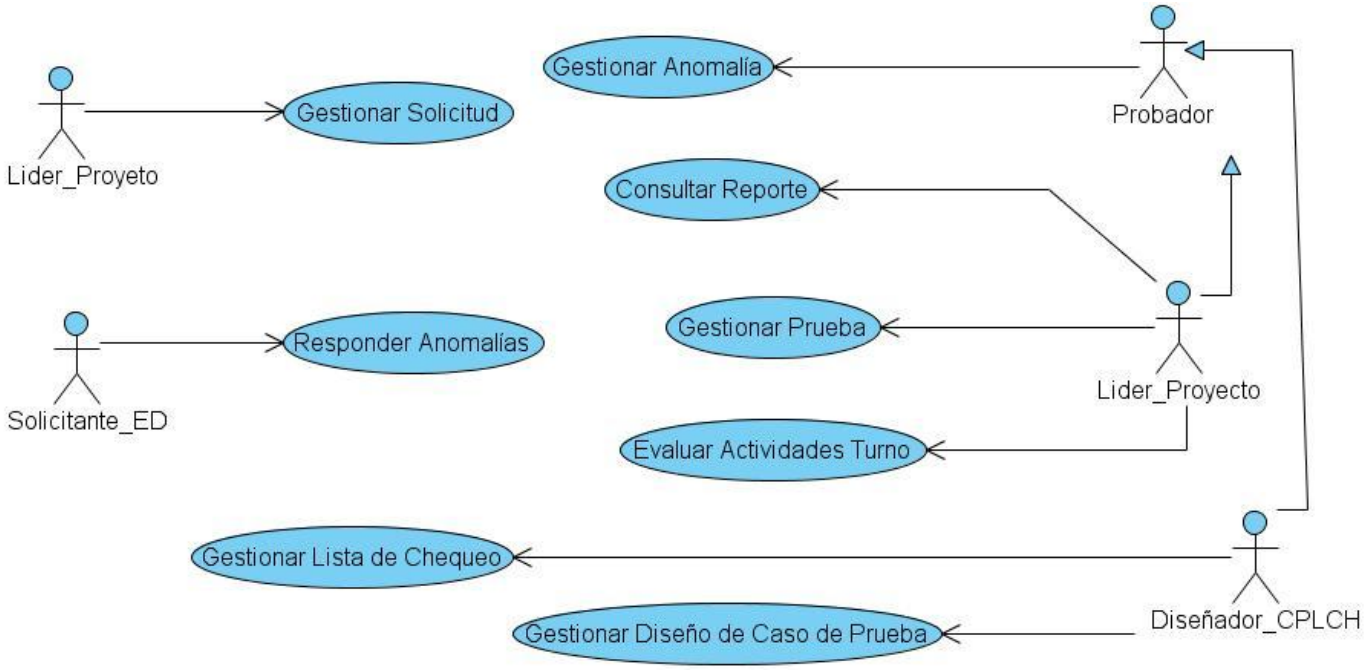


Figura 8.- Diagrama de Caso de Uso del Sistema

2.10 Casos de uso expandidos.

Tabla 27.- Descripción textual del caso de uso “Gestionar Prueba”.

Caso de uso	
CU-1	Gestionar Prueba
Propósito	Permite incluir, ver, modificar y eliminar las pruebas que se realizarán en el sistema.
Actores	Jefe_Prueba
Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Prueba. El actor puede incluir, ver, modificar y eliminar una Prueba. En caso de que seleccione la opción de incluir una prueba, el sistema da la posibilidad de introducir los datos necesarios para incluir una Prueba. Si el actor elige la opción de ver una prueba el sistema muestra el contenido de la prueba en cuestión. Si el actor elige la opción de modificar una Prueba., el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar un aprueba el sistema elimina la prueba en cuestión.

Tabla 28.- Descripción textual del caso de uso “Gestionar Anomalía”.

Caso de uso	
CU-3	Gestionar Anomalía
Propósito	Permite incluir, ver, modificar y eliminar anomalías.
Actores	Probador, Jefe_Prueba
Resumen:	El caso de uso inicia cuando el probador selecciona la opción que le permite realizar una acción sobre una Anomalía. El probador puede incluir, ver, modificar y eliminar una Anomalía. En caso de de que seleccione la opción de incluir una Anomalía, el sistema da la posibilidad de introducir los datos necesarios para incluir una Anomalía. Si el actor elige la opción de ver Anomalía el sistema muestra el contenido de dicha Anomalía. Si el actor selecciona la opción de modificar Anomalía. Si el actor selecciona la opción de eliminar una Anomalía el sistema elimina la anomalía en cuestión.

Tabla 29.- Descripción textual del caso de uso “Consultar Reporte”.

Caso de uso	
CU-7	Consultar Reporte
Propósito	Permite ver los reportes de anomalía por turno, los reportes de probador y los informes de anomalía por fecha.
Actores	Jefe_Prueba
Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un Reporte. El actor puede ver reporte de anomalía por turno, ver reporte probador, generar informe anomalía por fecha. En caso de que seleccione la opción de ver reporte de anomalía por turno, el sistema da la posibilidad de hacer el reporte correspondiente escogiendo el turno asociado. Si el actor elige la opción de ver reporte probador, el sistema muestra el reporte correspondiente escogiendo el probador asociado. Si el actor elige la opción de generar informe anomalía por fecha, el sistema muestra el informe de anomalía correspondiente asociado a la fecha escogida.

Las restantes descripciones de los casos de uso se pueden ver en el (**Anexo 1**).

2.11 Caso de Uso Arquitectónicamente Significativos

Los casos de uso aarquitectónicamente significativos son aquellos que ayudan a mitigar los riesgos más importantes, deben ser los más importantes para los usuarios del sistema y ayudan a cubrir las funcionalidades significativas

RF1- Gestionar Prueba.

La funcionalidad que responde a este requisito permite al jefe de prueba incluir las pruebas al sistema que se realizaran en toda la fase de prueba del proyecto, resulta muy importante lograr la correcta gestión de las pruebas a realizarse durante toda la fase de prueba a los proyectos.

RF3 - Gestionar Anomalía.

La funcionalidad que responde a este requisito permite al probador incluir las anomalías al sistema que se realizaran en toda la fase de prueba del proyecto. El probador gestionara las anomalías detectadas las cuales podrá modificar ver y eliminarlas en cualquier momento de acuerdo a sus permisos en el sistema.

Estas anomalías podrán ser gestionadas también por el jefe de prueba y el diseñador de casos de prueba.

RF6- Consultar Reporte.

La funcionalidad que responde a este requisito permite al jefe de prueba realizar los reportes pertinentes.

La realización de los reportes por parte del jefe de prueba ayuda mucho a la toma de decisiones y a la corrección de errores al Jefe de Prueba. También da la medida de cómo se efectúa el trabajo por parte de los probadores para poseer mayor visión para probar los productos a liberar dependiendo del nivel de envergadura. Además de servir para la toma de decisiones y la ejecución de acciones correctivas al jefe de prueba y tener un control estricto del esfuerzo de los probadores.

2.12 Conclusiones

Se abordaron las características principales del sistema mediante el desglose y análisis de los objetivos estratégicos y procesos de negocio haciendo un análisis crítico de cómo se ejecutan .Se describió el flujo actual de los eventos. Se describieron sistemas automatizados existentes. Por otra parte se detalló información manejable. Además se obtuvo la realización del modelo de negocio, y se especificaron los requisitos del software tanto funcionales como no funcionales y se definió los casos de uso del sistema.

Capítulo 3 Análisis y diseño del Sistema

3.1 Introducción

En este capítulo se realizan actividades referentes al análisis y diseño del sistema propuesto, el análisis y el diseño es un flujo de trabajo que tiene como propósito transformar los requisitos tanto funcionales como no funcionales en un diseño de clases concibiendo las relaciones e interacciones que presentan entre ellas. Se construyen los diagramas de clases del análisis y del diseño, se describen las clases y se diseña la base de datos.

3.2 Definición del modelo de análisis. Modelo de clases de análisis.

El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. **(EVA3, 2008).**

Transformar los requisitos en el Diseño del Futuro Sistema, no en el “Análisis del Futuro Sistema”, así que es de esperar que las actividades de análisis sean desarrolladas con el objetivo de facilitar la entrada al diseño, por lo que son un paso inicial y una primera aproximación conceptual para una vez comprendido los requisitos a este nivel, aumentar el nivel de especificidad en aras de garantizar el cubrimiento de los requisitos funcionales y no funcionales, considerando además el entorno de implementación (Lenguaje de programación, plataforma, Sistemas heredados con los cuales interactuar). **(EVA3, 2008).**

En el análisis se pueden estructurar los requisitos de manera que nos facilite su comprensión, su preparación, su modificación y en general su mantenimiento. Esta estructura (basada en clases de análisis y paquetes) es independiente de la estructura que se dio a los requisitos (basada en casos de uso). Sin embargo existe una trazabilidad directa entre esas distintas estructuras, la cual se define entre casos de uso del modelo de casos de uso y realizaciones del caso de uso en el modelo de análisis. **(EVA3, 2008).**

En el análisis existen las clases interfaz, entidad y controladora.

Análisis y Diseño del Sistema

Clase Interfaz: Modelan la interacción entre el sistema y sus actores.

Clase Entidad: Estas clases modelan información que posee una larga vida y que a menudo es persistente y fenómenos, conceptos y sucesos que ocurren en el mundo real.

Clase Control: Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

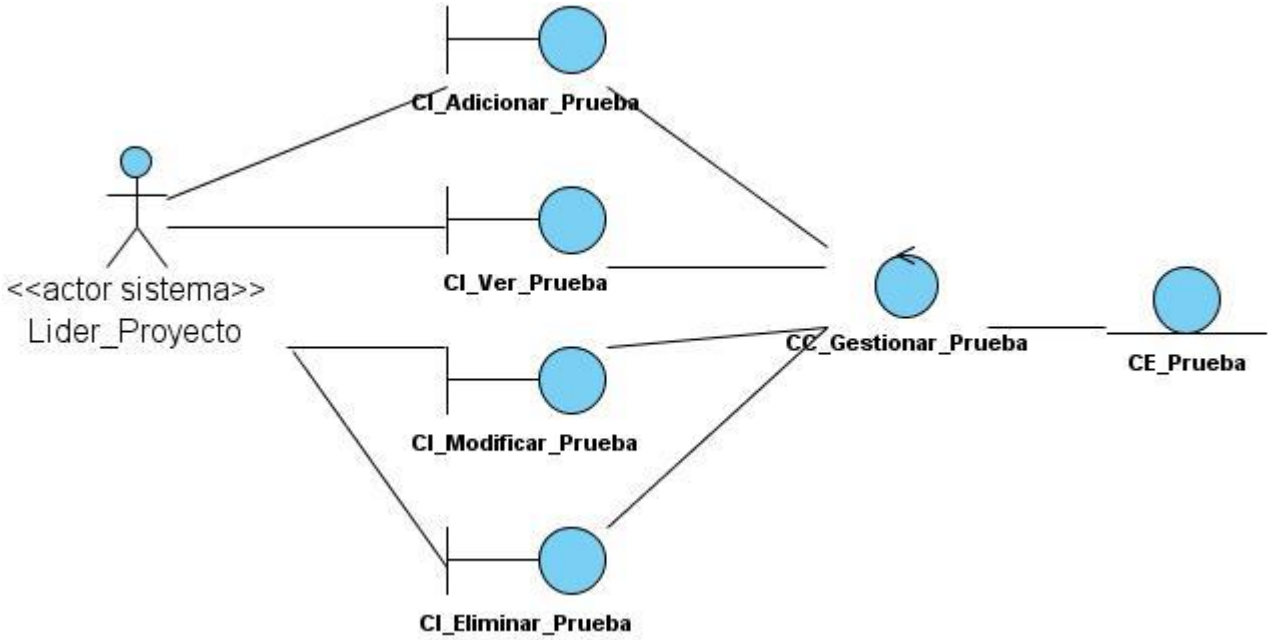


Figura 9.- Diagrama de clases de análisis del caso de uso “Gestionar Prueba”.

Análisis y Diseño del Sistema

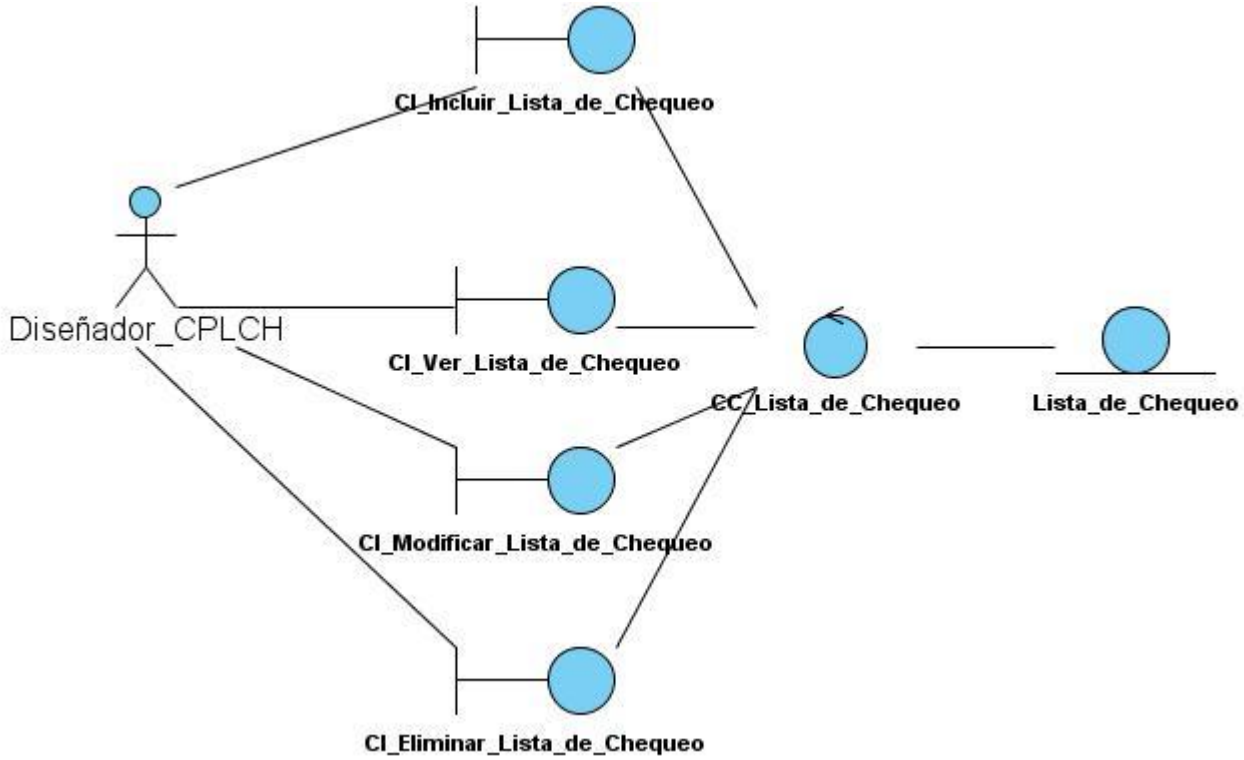


Figura 10.- Diagrama de clases de análisis del caso de uso “Gestionar Lista de Chequeo”.

Análisis y Diseño del Sistema

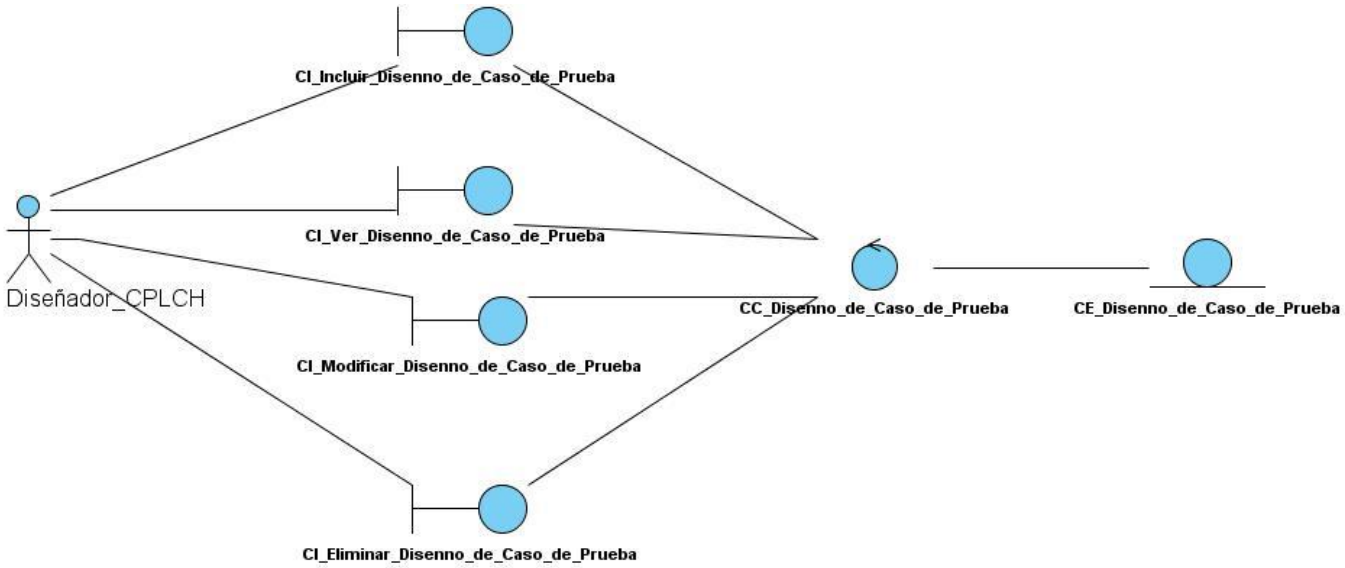


Figura 11.- Diagrama de clases de análisis del caso de uso “Gestionar Diseño de Caso de Prueba”

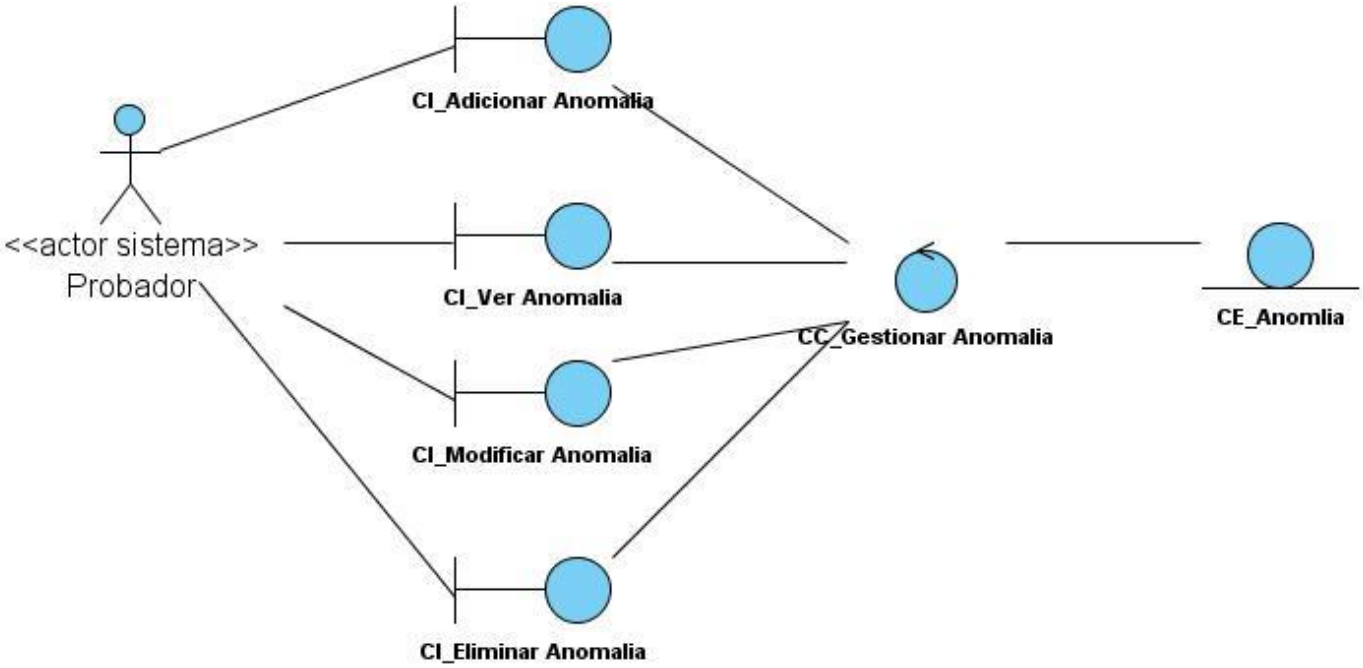


Figura 12.- Diagrama de clases de análisis del caso de uso “Gestionar Anomalía”.

Análisis y Diseño del Sistema

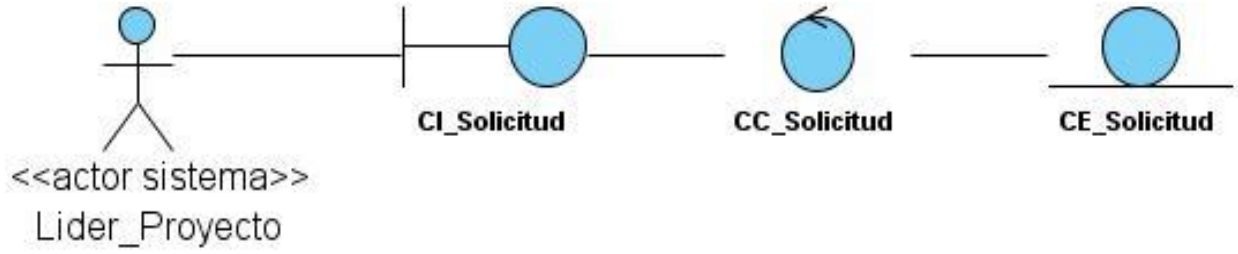


Figura 13.- Diagrama de clases de análisis del caso de uso “Gestionar Solicitud”.

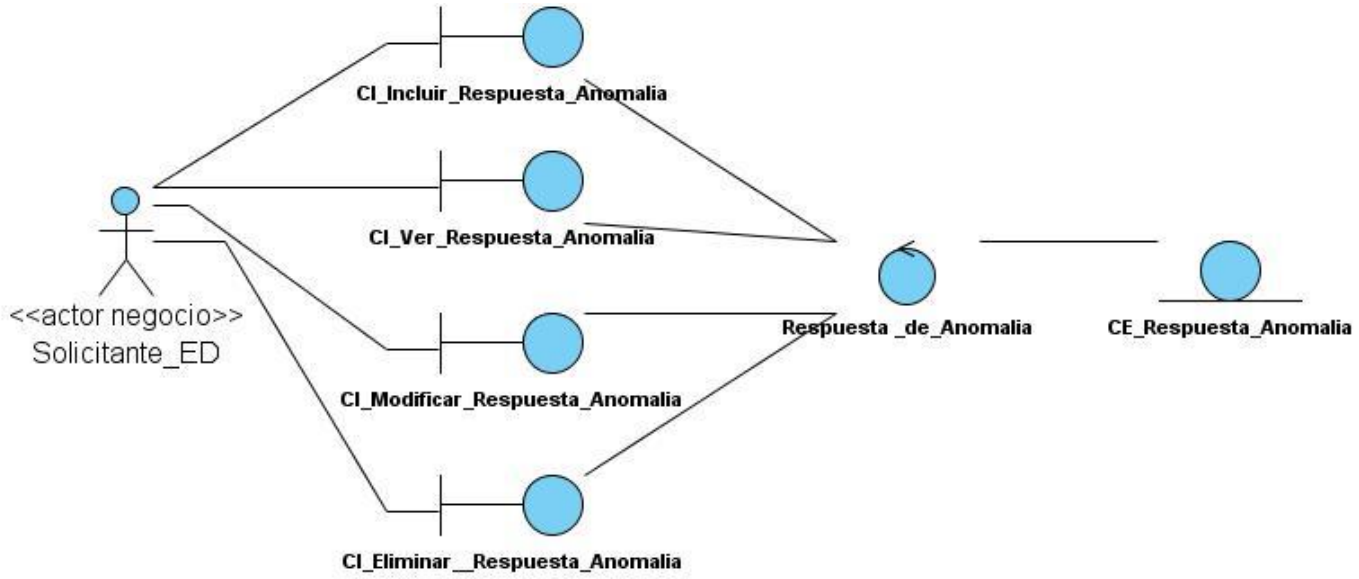


Figura 14.- Diagrama de clases de análisis del caso de uso “Gestionar Respuesta Anomalia”.

Análisis y Diseño del Sistema

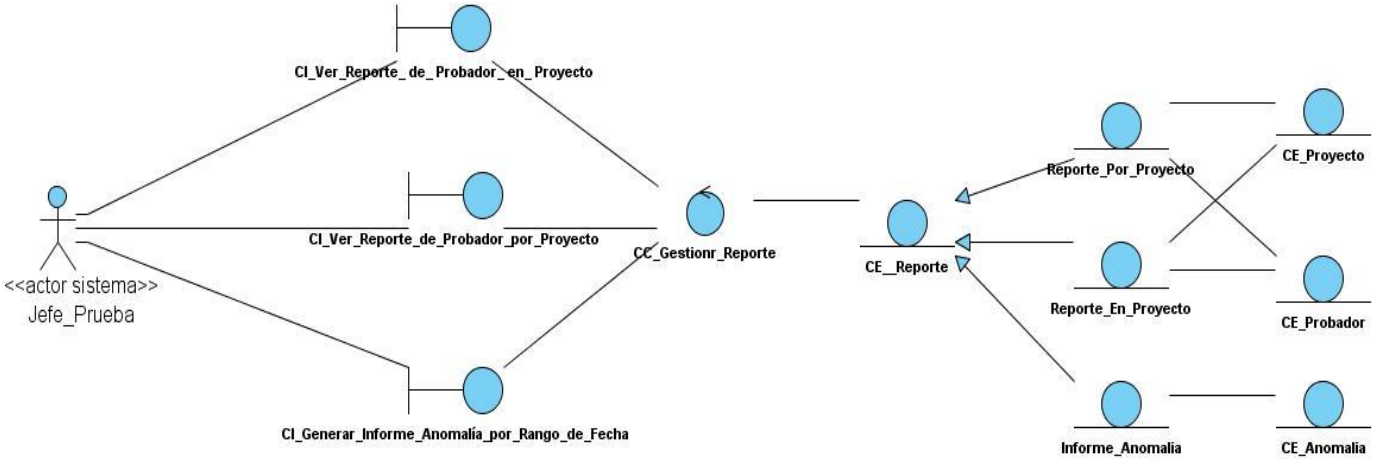


Figura 15.- Diagrama de clases de análisis del caso de uso “Consultar Reporte”.

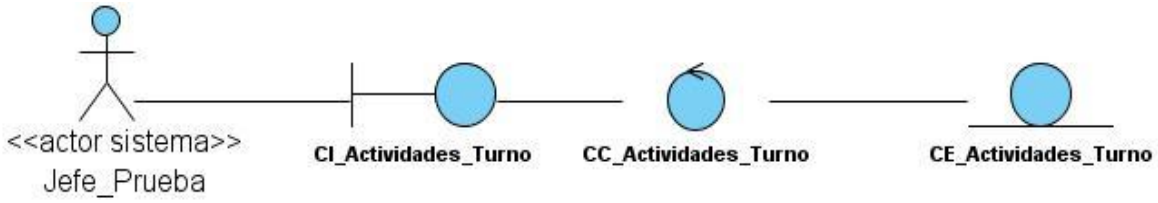


Figura 16.- Diagrama de clases de análisis del caso de uso “Evaluar Actividades Turno”.

Análisis y Diseño del Sistema

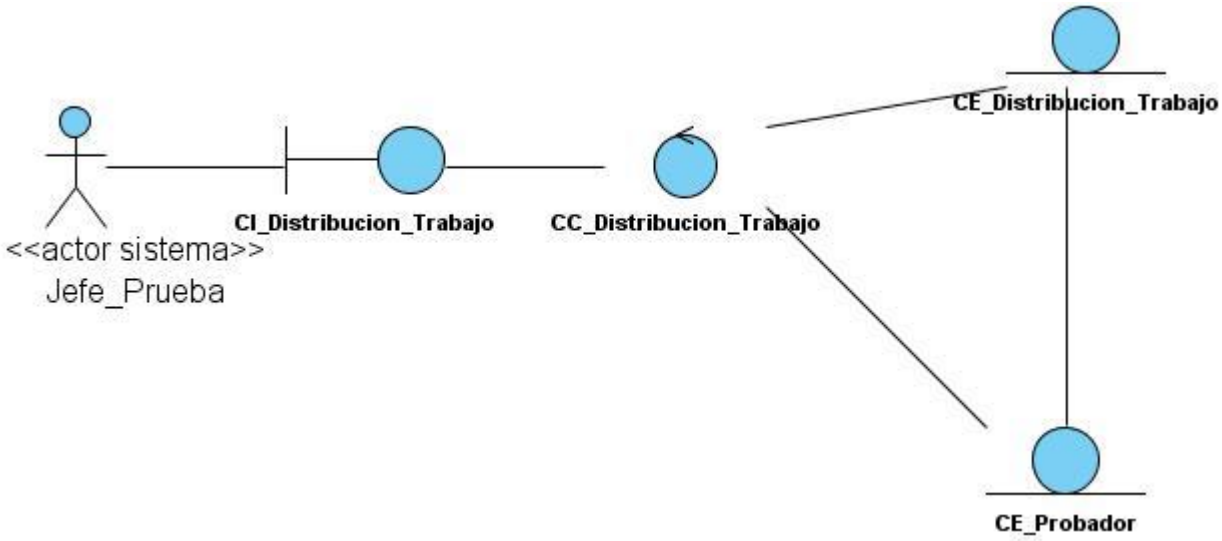


Figura 17.- Diagrama de clases de análisis del caso de uso “Evaluar Actividades Turno”.

3.3 Diagramas de Clases del Diseño con estereotipos Web.

Los diagramas de clases del diseño con estereotipo web mostrados representan a los caso de uso arquitectónicamente significativos., los cuales son Gestionar Prueba, Gestionar Anomalia y Consultar Reporte.

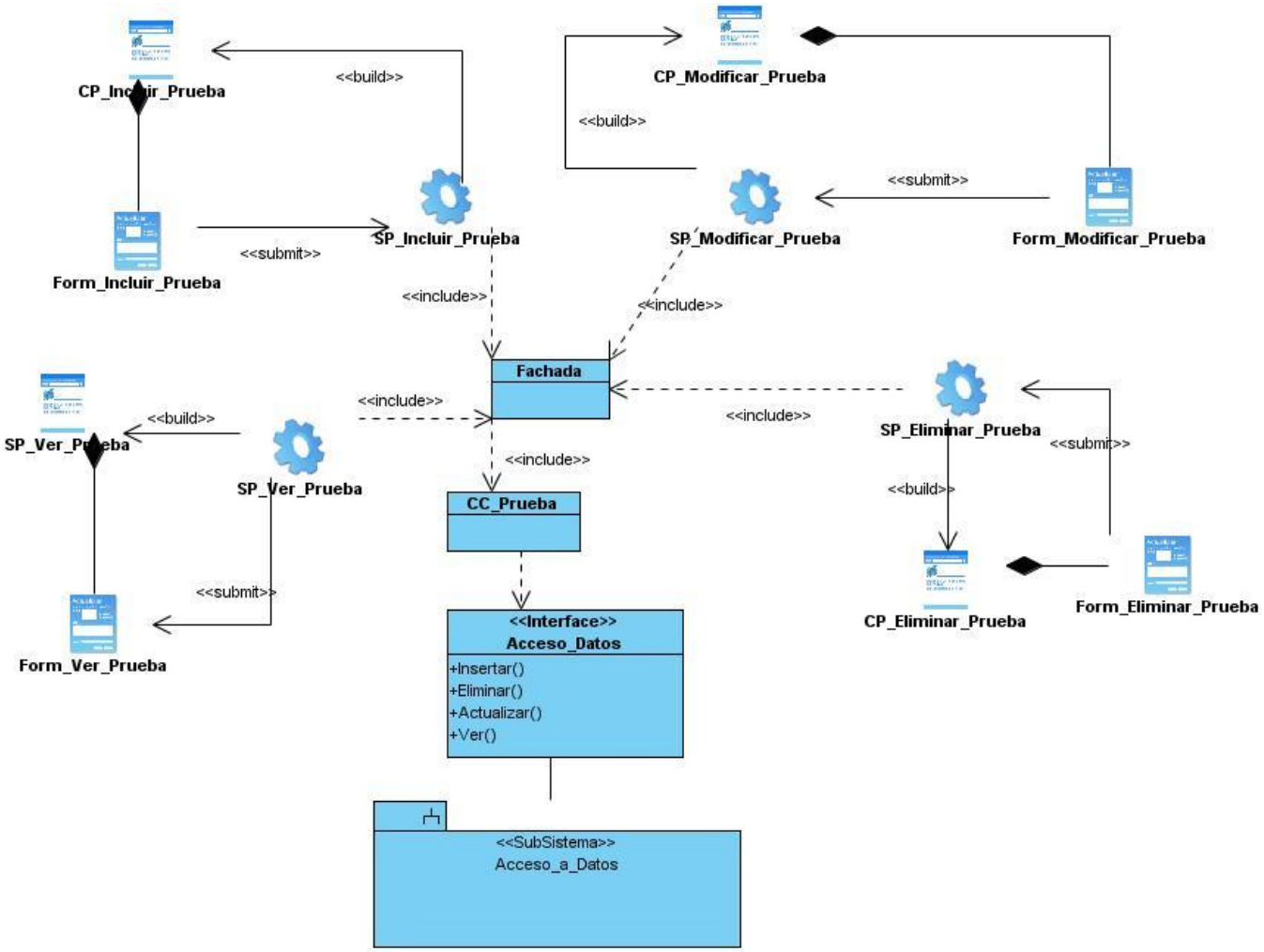


Figura 18.- Diagrama de clase del diseño del caso de uso “Gestionar Prueba”.

Análisis y Diseño del Sistema

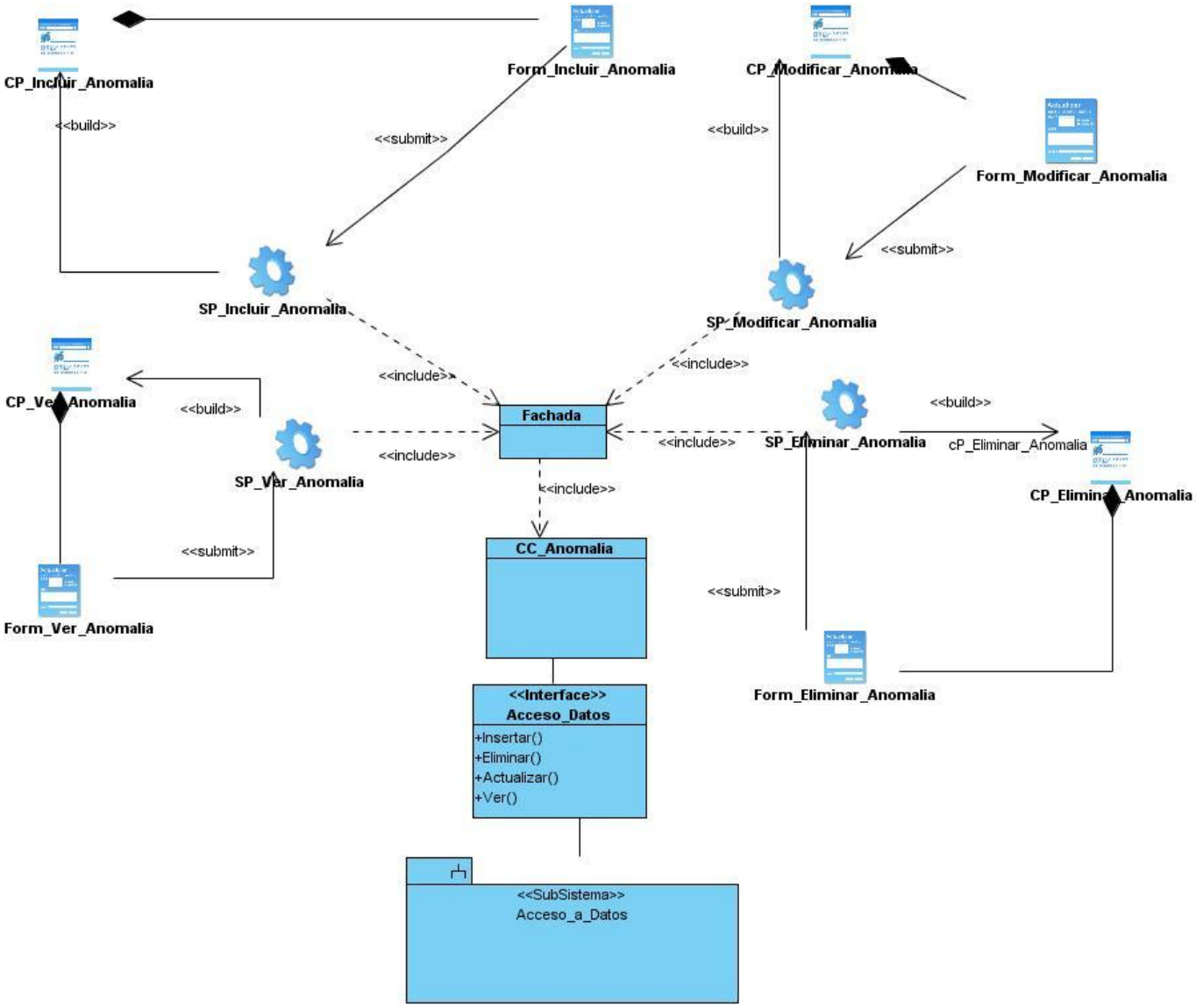


Figura 19.- Diagrama de clase del diseño del caso de uso “Gestionar Anomalía”.

Análisis y Diseño del Sistema

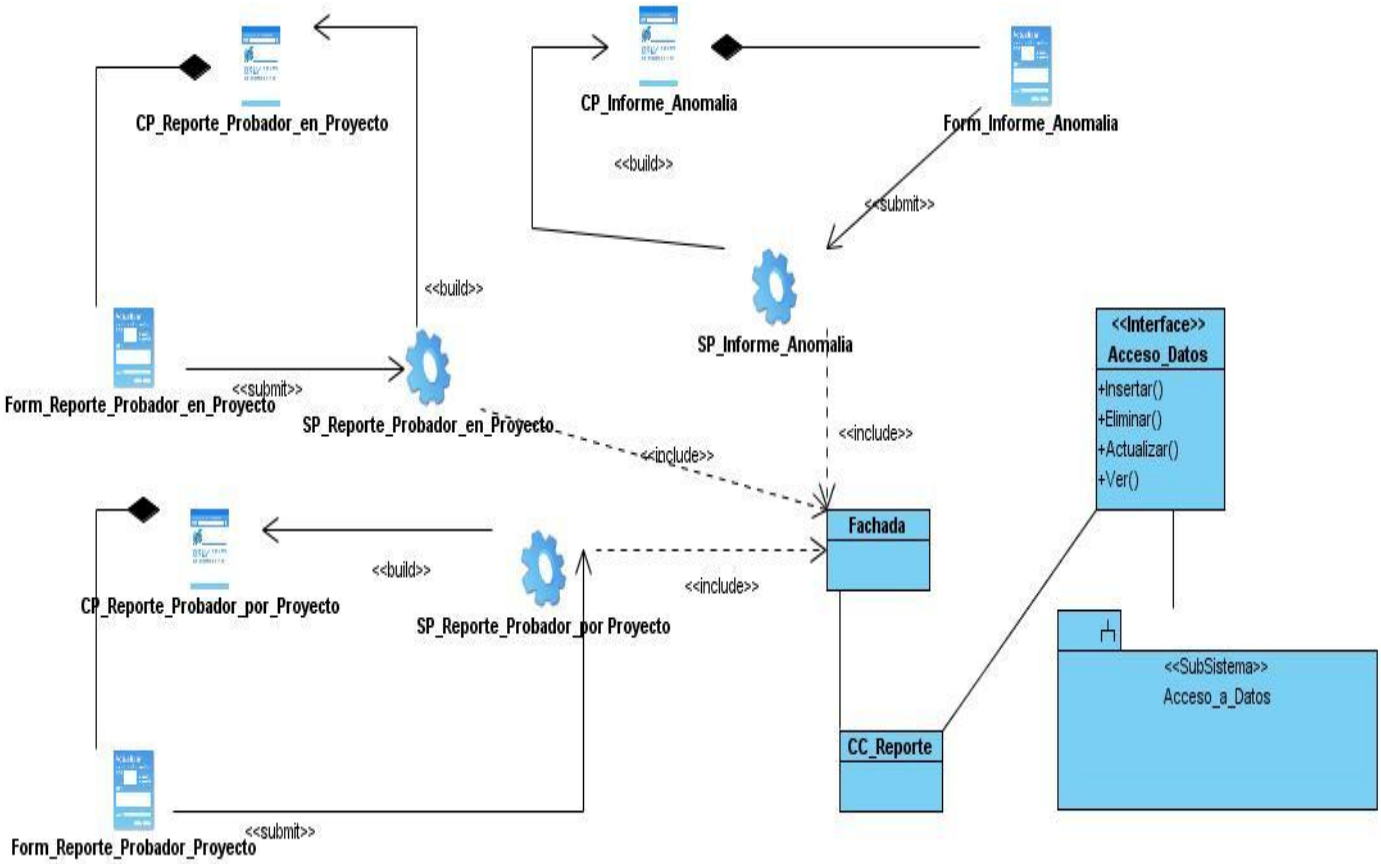


Figura 20.- Diagrama de clase del diseño del caso de uso "Consultar Reporte".

Los demás diagramas se representan en el (**Anexo 2**).

Análisis y Diseño del Sistema

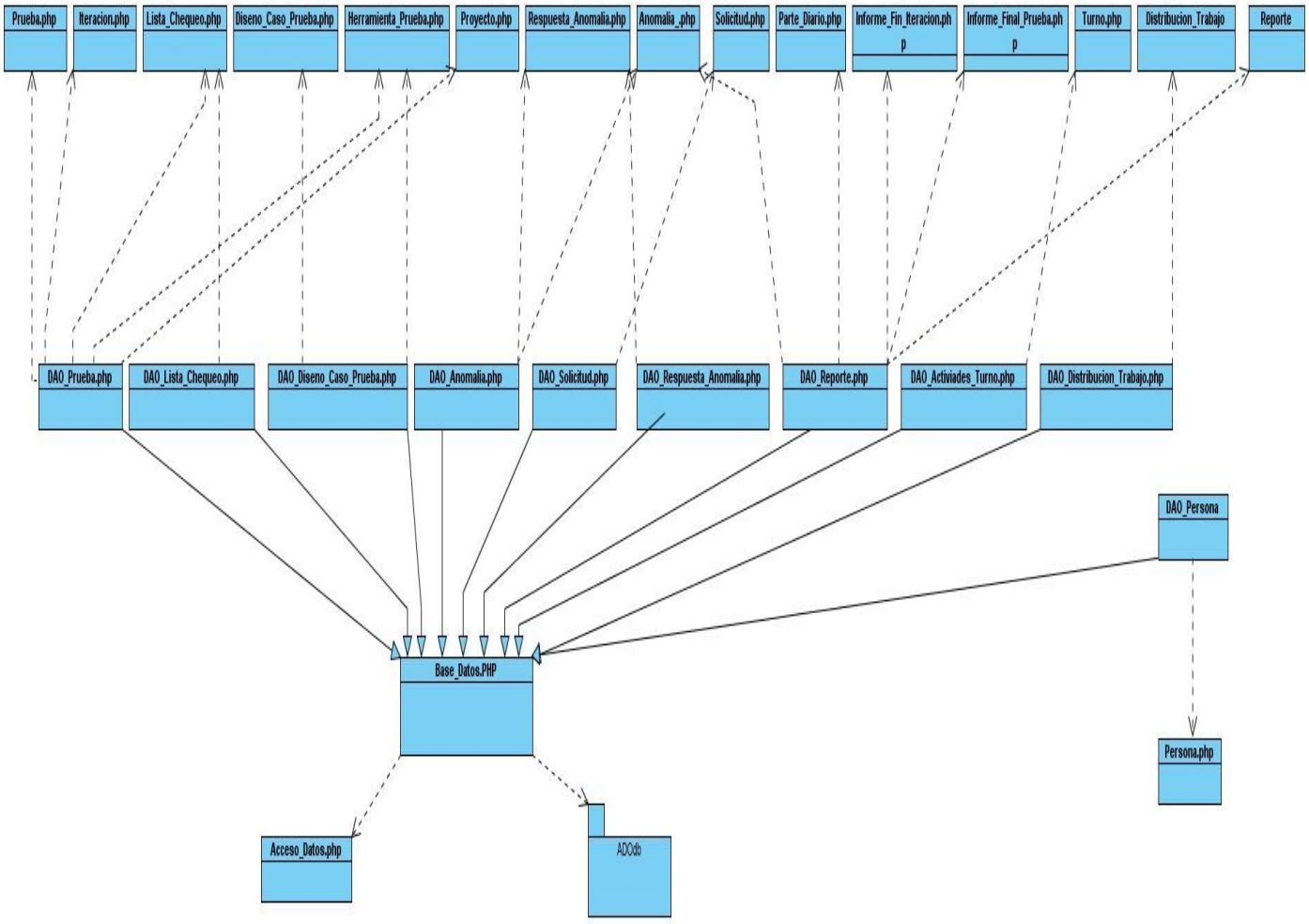


Figura 21.- Diagrama de clases del subsistema de Acceso a Datos.

3.4 Diagramas de Interacción.

La mayoría de los objetos que aparecen en un diagrama de interacción existirán mientras dure la interacción. Un diagrama de colaboración destaca la organización de los objetos que participan en una interacción y un diagrama de secuencia destaca la ordenación temporal de los mensajes.

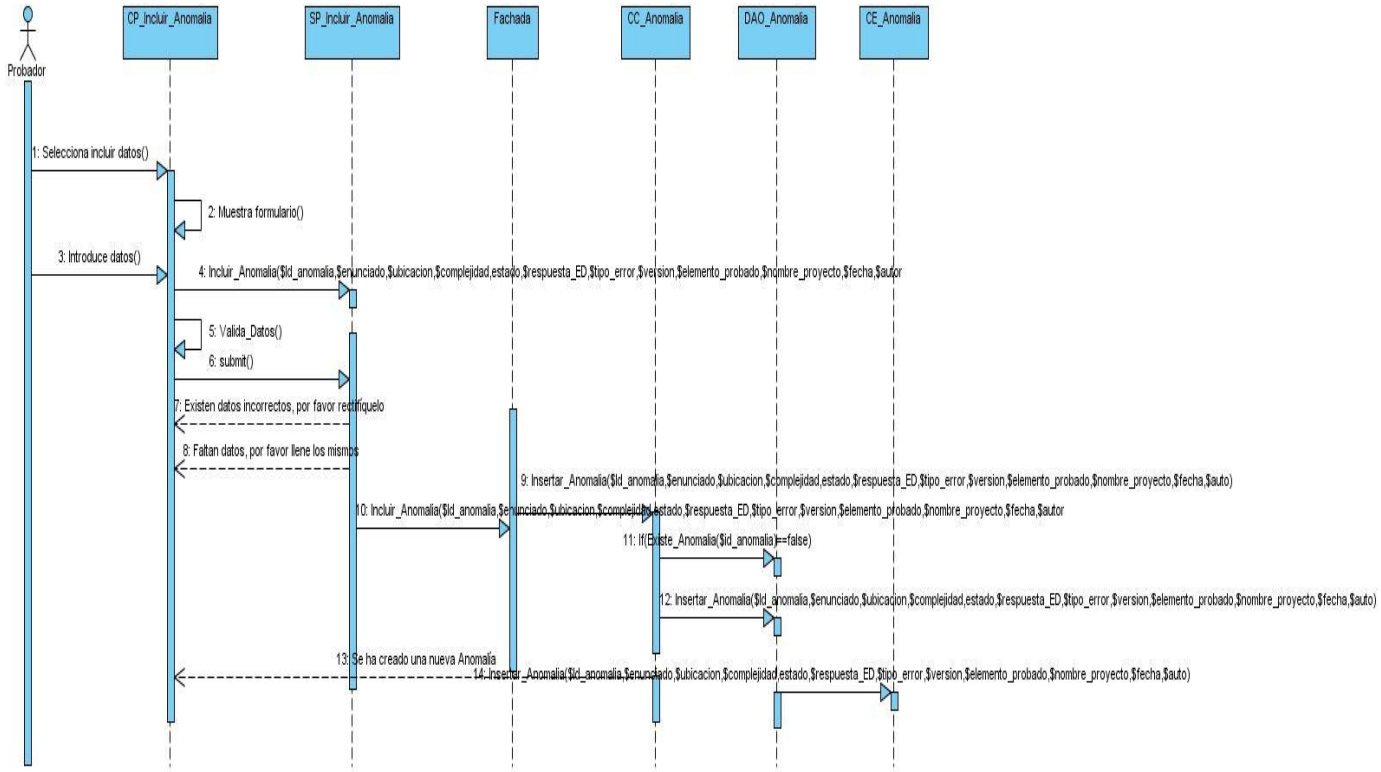


Figura 22.- Diagrama de secuencia del caso de uso “Gestionar Anomalía”. “Escenario Incluir Anomalía”.

Análisis y Diseño del Sistema

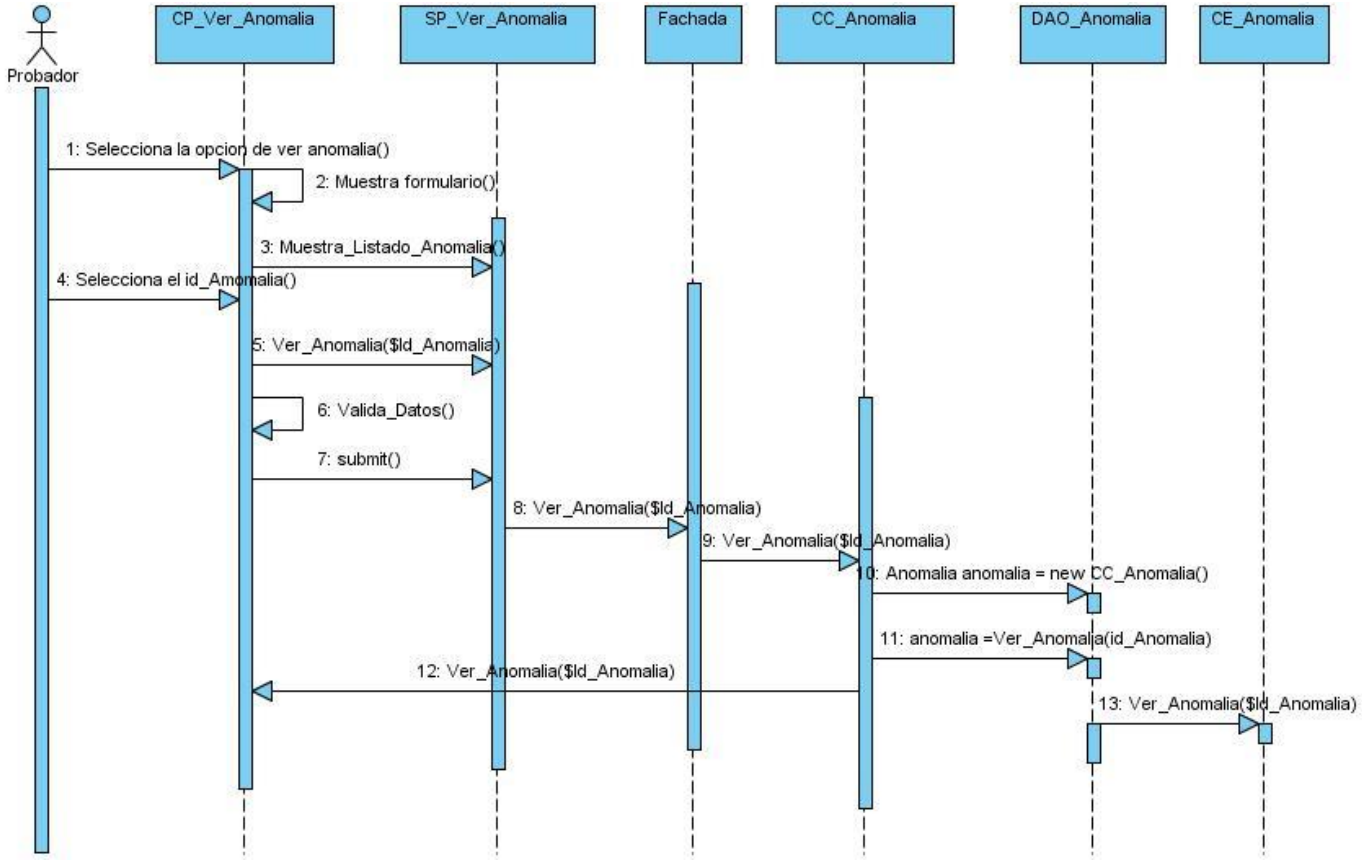


Figura 23.- Diagrama de secuencia del caso de uso “Gestionar Anomalía”. “Escenario Ver Anomalía”.

Análisis y Diseño del Sistema

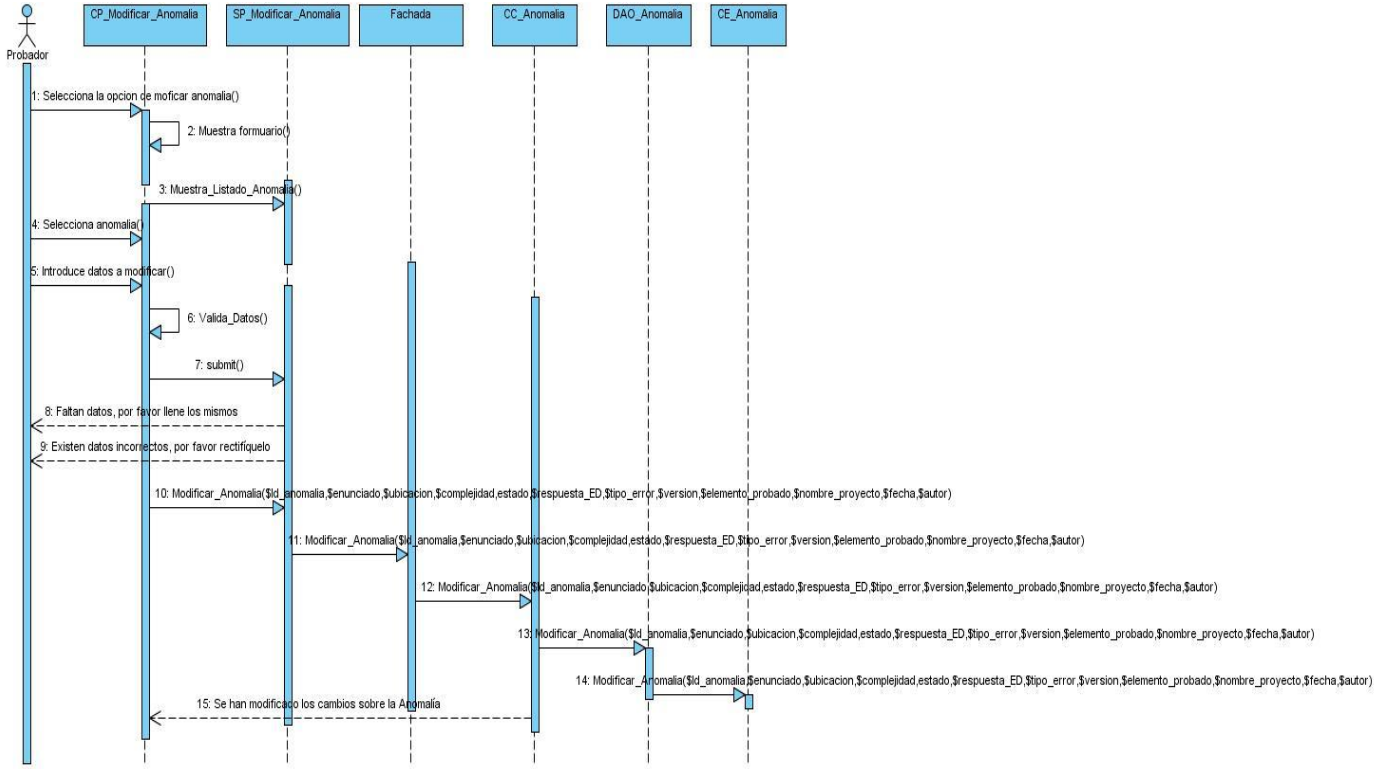


Figura 24.- Diagrama de secuencia del caso de uso “Gestionar Anomalía”.”Escenario Modificar Anomalía”.

Análisis y Diseño del Sistema

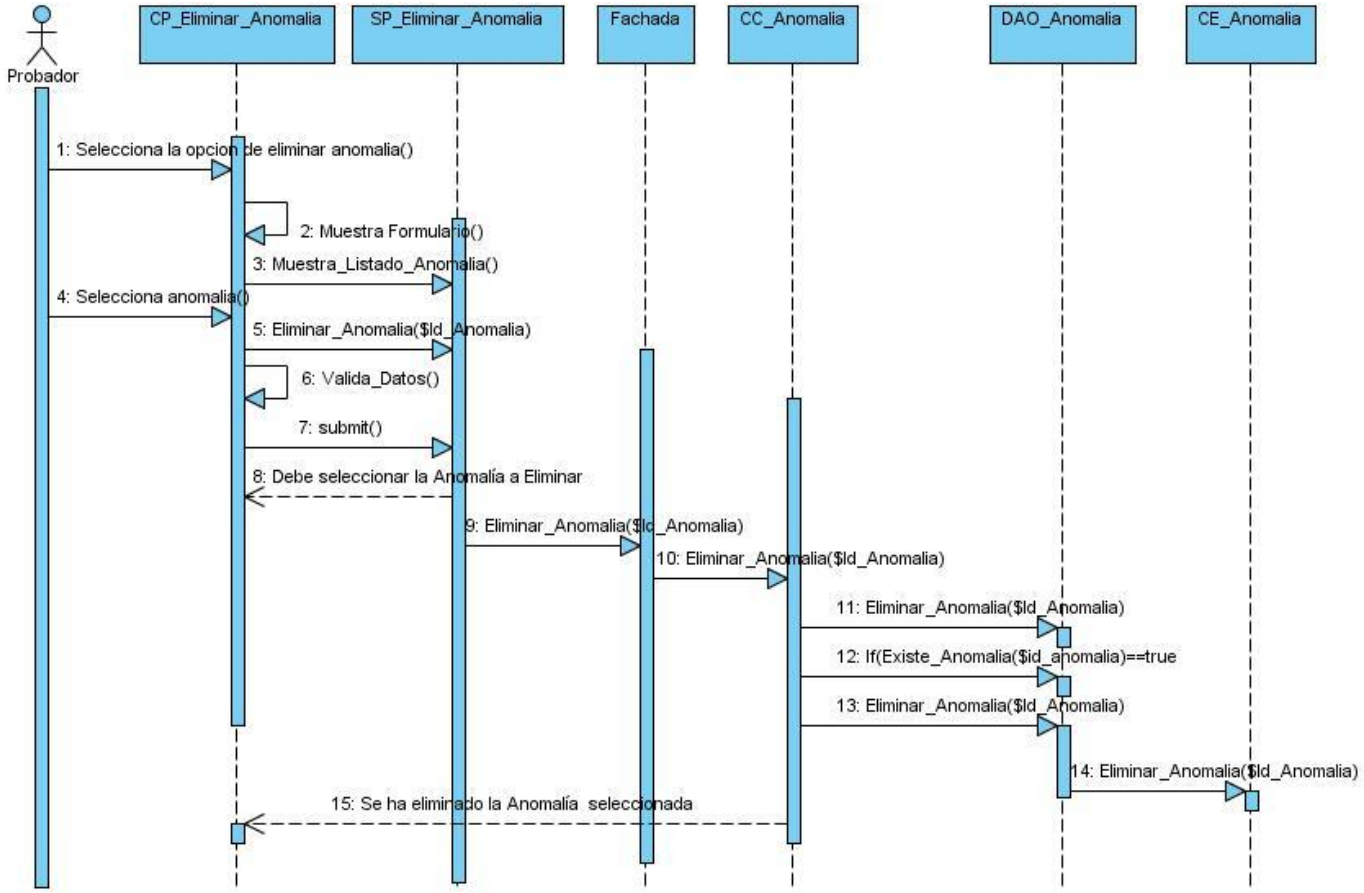


Figura 25.- Diagrama de secuencia del caso de uso “Gestionar Anomalía"."Escenario Eliminar Anomalía".

Análisis y Diseño del Sistema

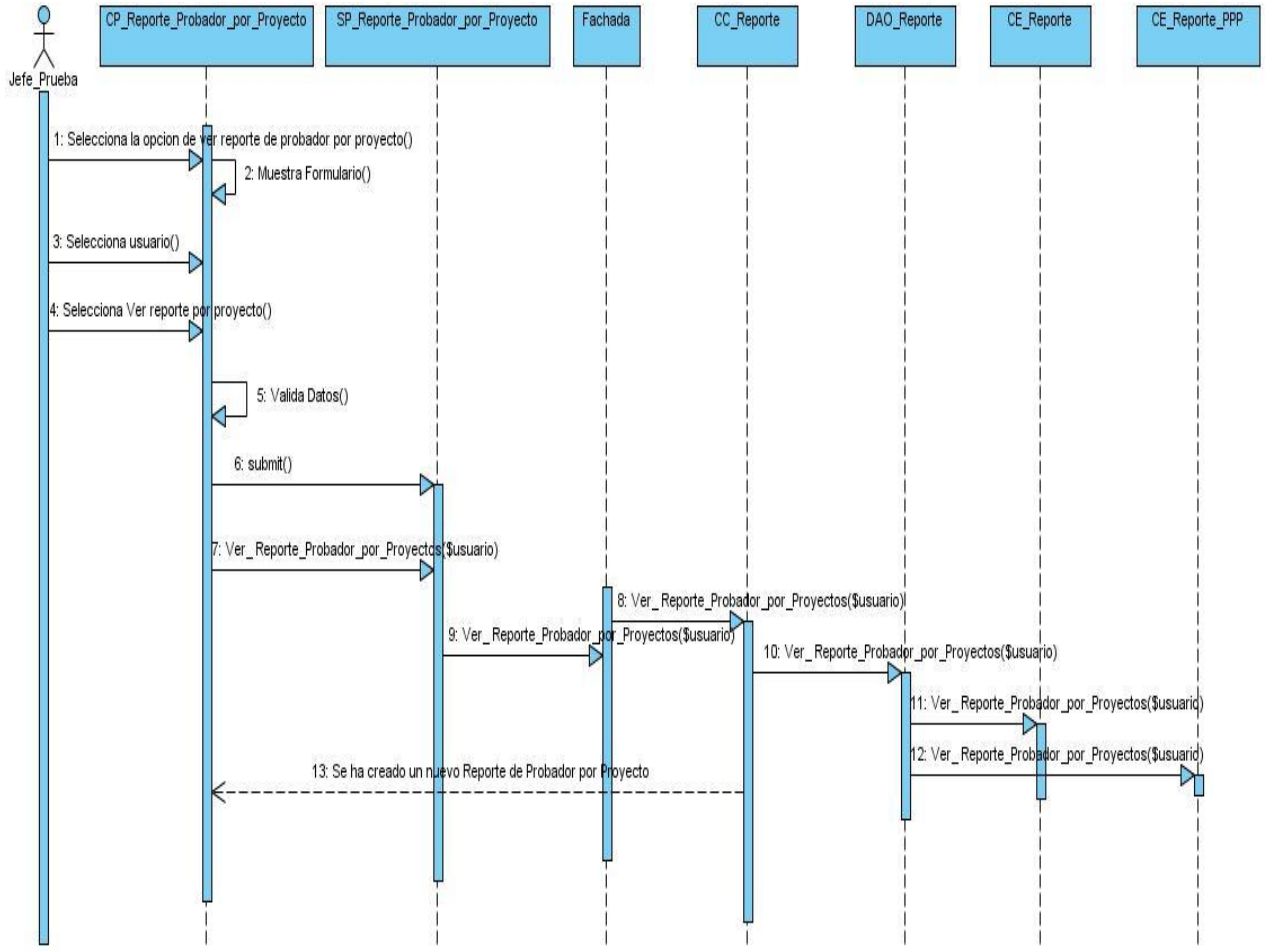


Figura 26.- Diagrama de secuencia del caso de uso “Consultar Reporte” “Escenario Ver Reporte de Probador por Proyecto”.

Análisis y Diseño del Sistema

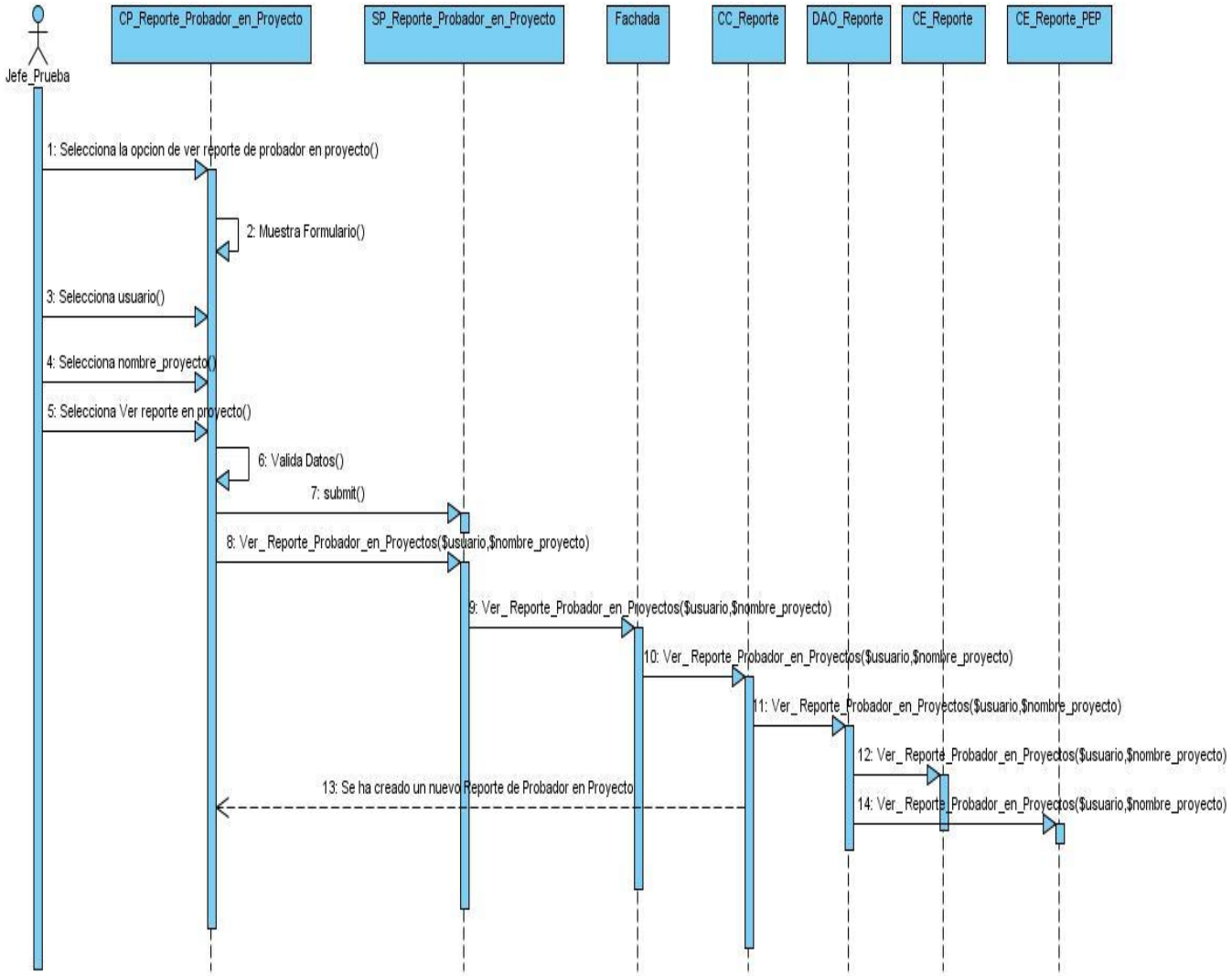


Figura 27.- Diagrama de secuencia del caso de uso “Consultar Reporte”. “Ver Reporte de Probador en Proyecto".

Análisis y Diseño del Sistema

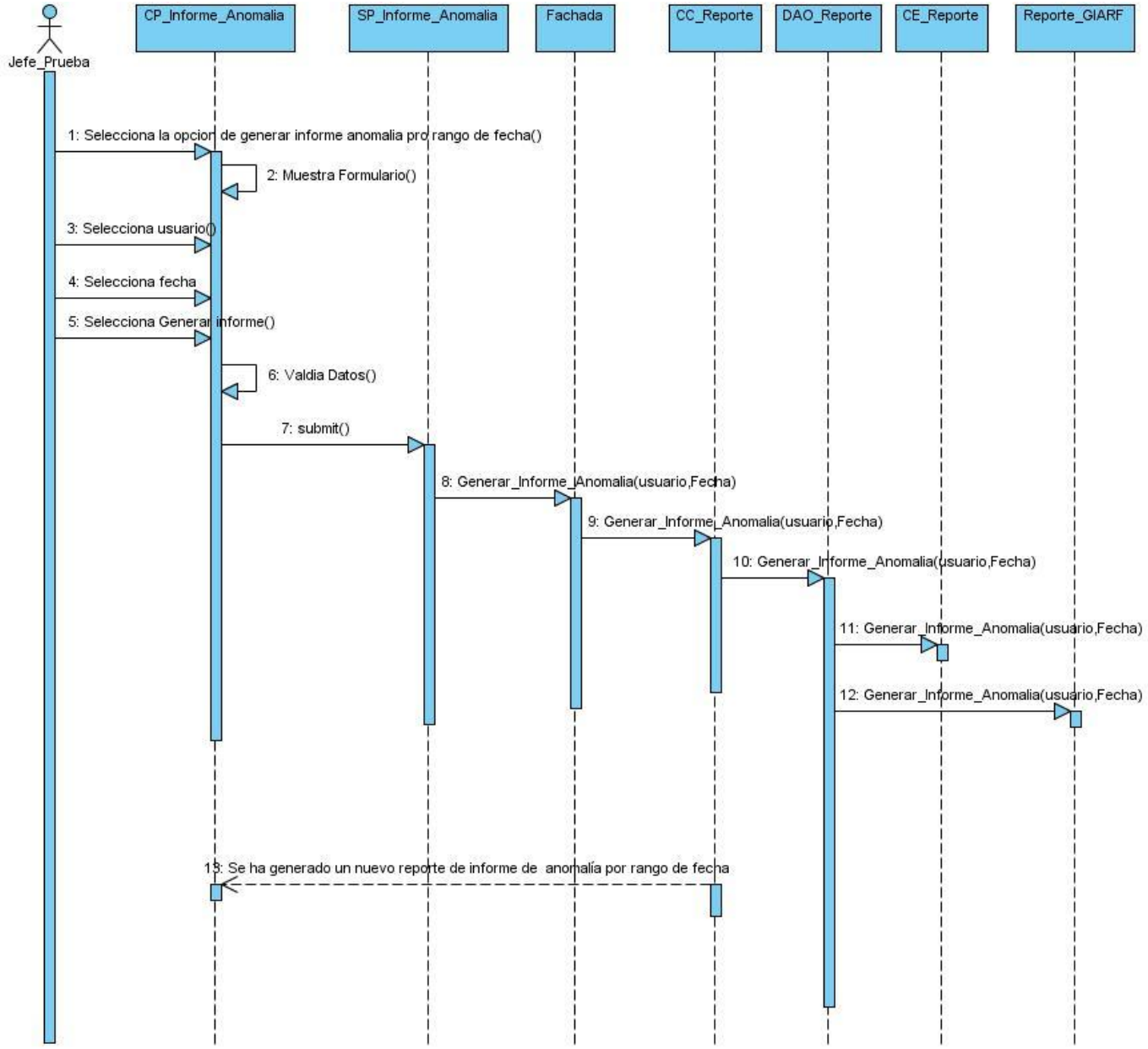


Figura 28.- Diagrama de secuencia del caso de uso “Consultar Reporte”. “Escenario Generar Informe. Informe Anomalía por Rango de Fecha”.

Para ver los restantes diagramas de interacción dirigirse al (Anexo 3).

Análisis y Diseño del Sistema

3.5 Diseño de la Base de Datos

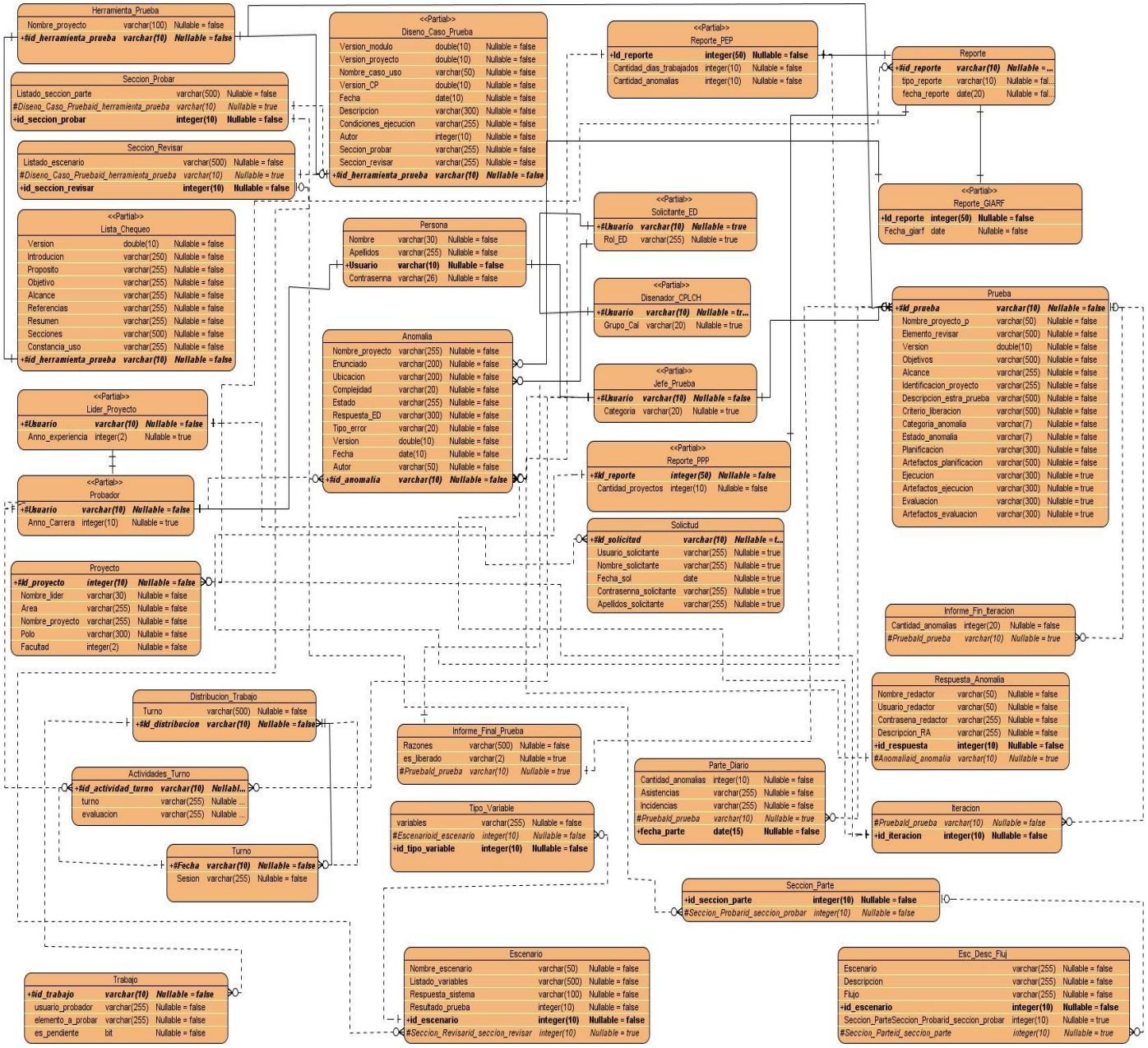


Figura 29.- Diagrama Entidad Relación de la Base de Datos.

Análisis y Diseño del Sistema

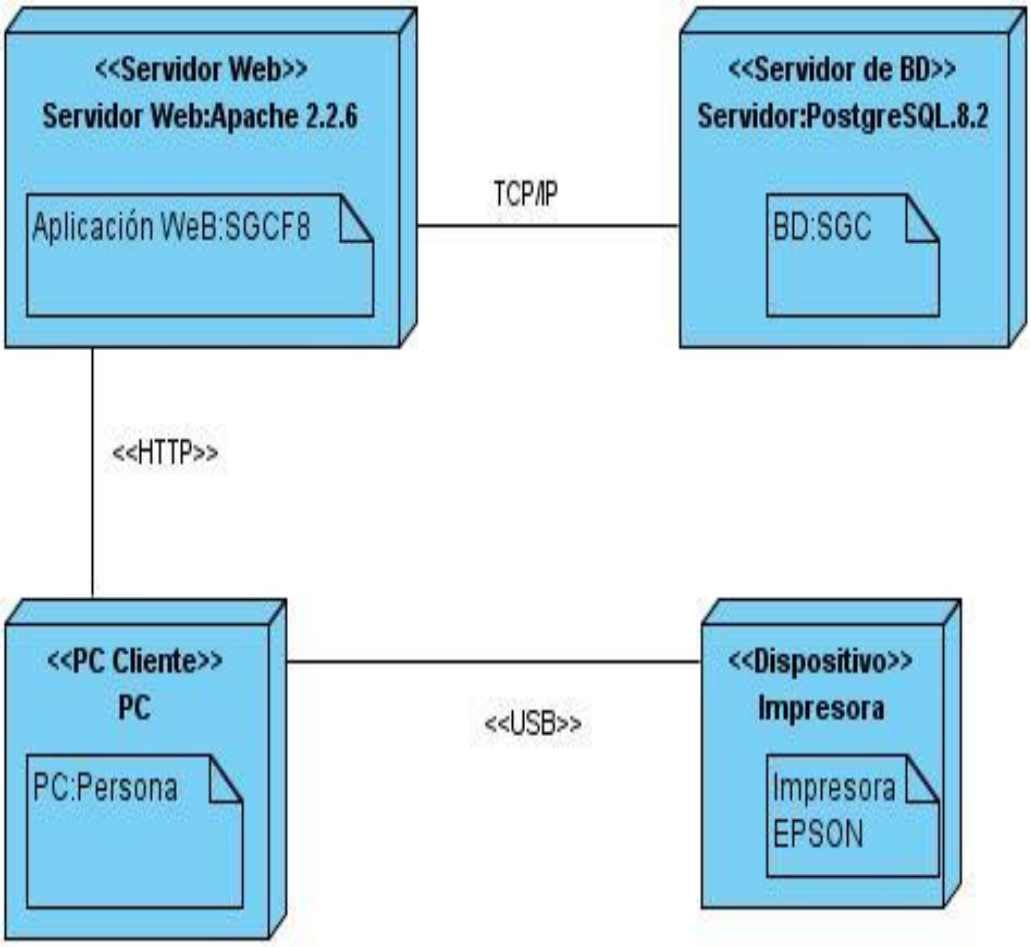


Figura 30.- Diagrama de Despliegue.

Análisis y Diseño del Sistema

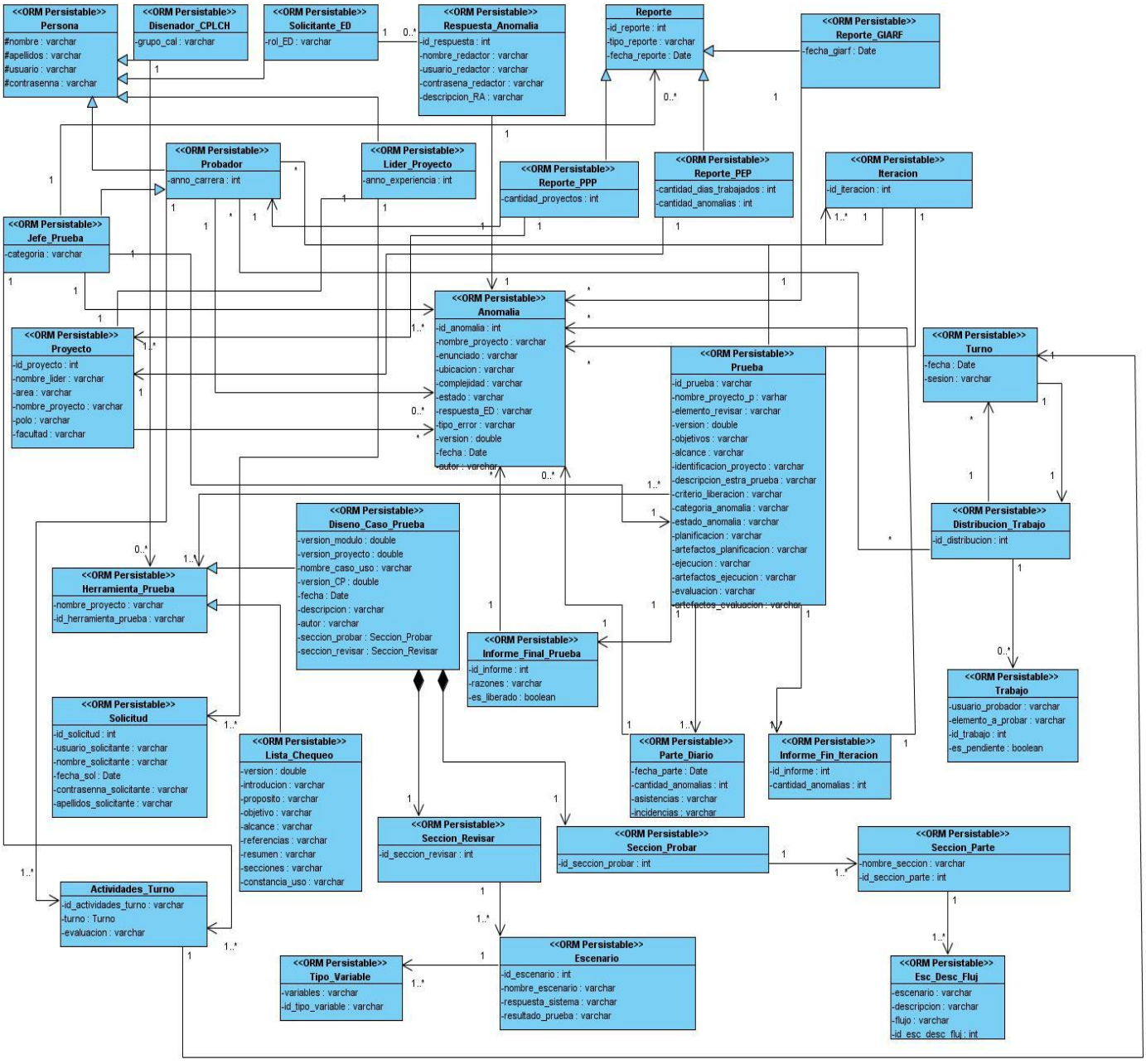


Figura 31. - Diagrama de Clases Persistentes.

Análisis y Diseño del Sistema

3.8 Descripción de las tablas

3.8.1 Clases entidades

Tabla 30. Clase entidad CE_Anomalia.

Nombre	CE_Anomalia	
Tipo de clase	Entidad	
Atributo		**Tipo**
Id_anomalia		int
enunciado		varchar
ubicacion		varchar
complejidad		varchar
estado		varchar
respuesta_ED		varchar
tipo_error		varchar
version		double
elemento_probado		varchar
nombre_proyecto		varchar
fecha		Date
autor		varchar
Responsabilidad		
Nombre	CE_Anomalia(pId_anomalia,penunciado,pubicacion,p complejidad,p estado,p respuesta_ED,ptipo_error,p version,p elemento_probado,p nombre_proyecto,pfecha,pautor)	
	GetId_Anomalia()	
	GetEnunciado()	

Análisis y Diseño del Sistema

	GetUbicacion()
	GetComplejidad()
	GetEstado()
	GetRespuesta()
	GetTipoError()
	GetVersion()
	GetElementoProbado()
	GetnombreProyecto()
	GetFecha()
	GetAutor()
Descripción	Clase entidad que representa la tabla Anomalía en la base de datos.

Tabla 31.- Clase Controladora CC_Anomalia

Nombre	CC_Anomalia
Tipo de clase	Controladora
Atributo	**Tipo**
Responsabilidad	
Nombre	Incluir_Anomalia(Id_anomalia, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor);
	Existe_Anomalia(id_anomalia)
	Cantidad_Anomalias_Dado_Nombre_Proyecto(nombre_proyecto)
	Obtener_Probador_de_Anomalia(id_anomalia)
	Obtener_Anomalias_Dado_Probador(usuario)

	Ver_Anomalia(Id_anomalía)
	Modificar_Anomalia(Id_anomalía, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor)
	Eliminar_Anomalia(id_anomalia)
Descripción	Controla las que se gestionan en el sistema por parte del probador y del jefe de proyecto.

Las restantes tablas de clases entidades y clases controladoras se muestran en el (**Anexo 4**).

Tabla 32.- Clase Interfaz Fachada

Nombre	Fachada	
Tipo de clase	Interfaz (PHP, servidor)	
Atributo		**Tipo**
control_prueba		CC_Prueba
control_anomalia		CC_Anomalia
control_resp_anomalia		CC_Respuesta_Anomalia
control_list_chequeo		CC_Lista_Chequeo
control_dis_caso_prueba		CC_Diseno_Caso_Prueba
control_reporte		CC_Reporte
control_act_turno		CC_Actividades_Turno
control_distri_trabajo		CC_Distribucion_Trabajo
Responsabilidad		
Nombre	Incluir_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion, categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion,	

Análisis y Diseño del Sistema

	artefactos_evaluacion);
	Existe_Prueba(nombre_proyecto)
	Existe_Prueba(id_prueba)
	Ver_Prueba(nombre_proyecto)
	Modificar_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion, categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion, artefactos_evaluacion)
	Eliminar_Prueba(nombre_proyecto)
	Incluir_Anomalia(Id_anomalia, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor);
	Existe_Anomalia(id_anomalia)
	Cantidad_Anomalias_Dado_Nombre_Proyecto(nombre_proyecto)
	Obtener_Probador_de_Anomalia(id_anomalia)
	Obtener_Anomalias_Dado_Probador(usuario)
	Ver_Anomalia(Id_anomalía)
	Modificar_Anomalia(Id_anomalía, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor)
	Eliminar_Anomalia(id_anomalia)
	Incluir_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_anomalia redactor,contraseña_redactor,descripción_RA);
	Existe_Respuesta_Anomalia(id_anomalia)
	Cantidad_Respuestas_Anomalias_Dado_Nombre_Proyecto (nombre_proyecto)
	Ver_Respuesta_Anomalia(Id_respuesta)
	Modificar_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_ redactor,contraseña_redactor,descripción_RA)
	Eliminar_Respuesta_Anomalia(id_anomalia)

Análisis y Diseño del Sistema

	Incluir_Lista_Chequeo(id_herramienta_prueba,nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)
	Ver_Lista_Chequeo(id_herramienta_prueba)
	Modificar_Lista_Chequeo(id_herramienta_prueba,nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)
	Buscar_Lista_Chequeo(id_herramienta_prueba)
	Eliminar_Lista_Chequeo(id_herramienta_prueba)
	Existe_ Lista_Chequeo(id_herramienta_prueba)
	Incluir_ Diseno_Caso_Prueba (id_herramienta_prueba ,nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)
	Ver_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Modificar_Diseno_Caso_Prueba (nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)
	Buscar_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Eliminar_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Existe__ Diseno_Caso_Prueba(id_herramienta_prueba)
	Ver_ Reporte_Probador_por_Proyectos(usuario)
	Cantidad_Proyectos_Probados(usuario)
	Ver_ Reporte_Probador_en_Proyectos(usuario,nombre_proyecto)
	Buscar_Anomalias_Probador(nombre_proyecto,usuario)
	Buscar_Anomalias_Probador_por_Rango_Fecha(nombre_proyecto,usuario,rango_fecha)
	Cantidad_anomalias(nombre_proyecto,usuario,rango_fecha)
	Cantidad_Dias_Trabajados(nombre_proyecto,usuario,rango_fecha)
	Generar_Informe_Anomalia(usuario,ranfo_fecha)
	Buscar_Probador_Por_Turno(usuario)

Análisis y Diseño del Sistema

	Listado_Probador_Por_Turno(turno)
	Evaluar_Probador(turno)
	Dar_Evaluacion(evaluacion,usuario)
	Incluir_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contrasenna_solicitante, nombre_proyecto, fecha_solicitud);
	Existe_Solicitud(id_solicitud)
	Obtener_Solicitanter(id_solicitud)
	Ver_Solicitud(id_solicitud)
	Modificar_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contrasena_solicitante, nombre_proyecto, fecha_solicitud)
	Eliminar_Solicitud(id_solicitud)
	Existe_Solicitud(id_solicitud)
	Obtener_Distribucion(id_distribucion)
	Ver_Trabajo_Pendiente()
	Asignar_Trabajo_Pendiente()
	Buscar_Trabajo()
	Mostrar_Iteracion_Prueba()
	Listar_CasoUso_Probados()
	Listar_CasoUso_No_Probados()
Descripción	Controla la interacción entre todas las clases del sistema.

Tabla 33. - Clase DAO_Prueba

Nombre	DAO_Prueba
Tipo de clase	DAO
Atributo	**Tipo**

Análisis y Diseño del Sistema

Responsabilidad	
Nombre	Incluir_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion, categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion, artefactos_evaluacion);
	Existe_Prueba(nombre_proyecto)
	Ver_Prueba(nombre_proyecto)
	Modificar_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion, categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion, artefactos_evaluacion)
	Eliminar_Prueba(nombre_proyecto)
Descripción	Controla las pruebas que se gestionan en el sistema por parte del jefe de proyecto.

Tabla 34.- Clase DAO_Anomalia.

Nombre	DAO_Anomalia
Tipo de clase	DAO
Atributo	**Tipo**
Responsabilidad	
Nombre	Incluir_Anomalia(Id_anomalia, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor);

Análisis y Diseño del Sistema

	Existe_Anomalia(id_anomalia)
	Cantidad_Anomalias_Dado_Nombre_Proyecto(nombre_proyecto)
	Obtener_Probador_de_Anomalia(id_anomalia)
	Obtener_Anomalias_Dado_Probador(usuario)
	Ver_Anomalia(Id_anomalía)
	Modificar_Anomalia(Id_anomalía, enunciado, ubicación,complejidad,estado, respuesta_ED, tipo_error, version, elemento_probado, nombre_proyecto,fecha,autor)
	Eliminar_Anomalia(id_anomalia)
Descripción	Controla las que se gestionan en el sistema por parte del probador y del jefe de proyecto.

Tabla 35. - Clase DAO_Respuesta_Anomalia.

Nombre	DAO_Respuesta_Anomalia	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_ redactor,contraseña_redactor,descripción_RA);	
	Existe_Respuesta_Anomalia(id_anomalia)	
	Cantidad_Respuestas_Anomalias_Dado_Nombre_Proyecto (nombre_proyecto)	
	Ver_Respuesta_Anomalia(Id_respuesta)	
	Modificar_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_ redactor,contraseña_redactor,descripción_RA)	

Análisis y Diseño del Sistema

	Eliminar_Respuesta_Anomalia(id_anomalia)
Descripción	Controla las respuestas de anomalías que puede realizar el solicitante del equipo de desarrollo.

Tabla 36.- Clase DAO_Lista_Chequeo.

Nombre	DAO_Lista_Chequeo	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Lista_Chequeo(nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)	
	Ver_Lista_Chequeo(nombre_proyecto)	
	Modificar_Lista_Chequeo(nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)	
	Buscar_Lista_Chequeo(nombre_proyecto)	
	Eliminar_Lista_Chequeo (nombre_proyecto)	
Descripción	Controla las listas de chequeo de los proyectos en prueba.	

Análisis y Diseño del Sistema

Tabla 37.- Clase DAO_Diseno_Caso_Prueba

Nombre	DAO_Diseno_Caso_Prueba	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_ Diseno_Caso_Prueba (nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)	
	Ver_ Diseno_Caso_Prueba (nombre_proyecto)	
	Modificar_Diseno_Caso_Prueba (nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)	
	Buscar_ Diseno_Caso_Prueba (nombre_proyecto)	
	Eliminar_ Diseno_Caso_Prueba (nombre_proyecto)	
Descripción	Controla los diseños de caso de prueba de los proyectos en prueba.	

Tabla 38.- Clase DAO_Reporte.

Nombre	DAO_Reporte	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		

Análisis y Diseño del Sistema

Nombre	Ver_ Reporte_Probador (usuario)
	Buscar_Anomalias_Probador(nombre_proyecto,iteración,usuario)
	Ver_Reporte_Anomalia_Turno(turno)
	Generar_Informe_Anomalia(Fecha)
	Generar_Informe_Anomalia(iteracion)
	Generar_Informe_Anomalia(nombre_proyecto)
	Generar_Informe_Anomalia(rango_fecha)
Descripción	Controla los reportes que se pueden hacer en el sistema por parte de jefe de prueba.

Tabla 39.- Clase DAO_Activiades_Turno.

Nombre	DAO_Activiades_Turno	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Buscar_Probador_Por_Turnob(usuario)	
	Listado_Probadorpor_Turno(turno)	
	Evaluar_Probador(turno)	
	Dar_Evaluacion(evaluacion,usuario)	
Descripción	Controla los reportes que se pueden hacer en el sistema por parte de jefe de prueba.	

Análisis y Diseño del Sistema

Tabla 40. - Clase DAO_Solicitud.

Nombre	DAO_Solicitud	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contraseña_solicitante, nombre_proyecto, fecha_solicitud);	
	Existe_Solicitud(id_solicitud)	
	Obtener_Solicitanter(id_solicitud)	
	Ver_Solicitud(id_solicitud)	
	Modificar_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contrasena_solicitante, nombre_proyecto, fecha_solicitud)	
	Eliminar_Solicitud(id_solicitud)	
Descripción	Controla las solicitudes que se gestionan en el sistema por parte del solicitante del equipo de desarrollo.	

Tabla 41.- Clase DAO_Distribucion_Trabajo.

Nombre	DAO_Distribucion_Trabajo	
Tipo de clase	DAO	
Atributo		**Tipo**
Responsabilidad		
Nombre	Ver_Trabajo_Pendiente()	

	Asignar_Trabajo_Pendiente()
	Buscar_Trabajo()
	Mostrar_Iteracion_Prueba()
	Listar_CasoUso_Probados()
	Listar_CasoUso_No_Probados()
Descripción	Controla las solicitudes que se gestionan en el sistema por parte del solicitante del equipo de desarrollo.

3.9 Definiciones de diseño que se apliquen.

La utilización de un buen diseño en los sistemas es uno de los factores decisivos en la aceptación por parte de los clientes en el uso del software, la interfaz de un sistema, y la salido de los datos en un buen estado conllevan a esta aceptación, por otra parte la ayuda y el tratamiento de excepciones o errores guían al usuario por un camino seguro y tienen gran influencia en lograr atraer o alejar al cliente.

3.10 Tratamiento de errores.

El tratamiento de los errores de un sistema es aspecto influye en su correcto funcionamiento, con el correcto tratamiento de los errores se garantiza la integridad de la información. Para lograr esto, fueron previstos todos los errores posibles que pudiera generar el sistema a partir de la interacción del usuario.

3.11 Seguridad.

Un sistema se considera seguro cuando sus datos no están en riesgo, para lograr esto todos los usuarios deben tener su determinado permiso para no poner en riesgo el sistema. Cada usuario tendrá una contraseña para su autenticación y el posterior acceso a las funcionalidades permitidas.

3.12 Interfaz.

La interfaz de usuario dará posibilidad de que el usuario se relacione con el sistema. Cada detalle de la interfaz será vista por el usuario para su interacción y así lograr el objetivo de esta interacción. Las

interfaces deben ser capaces de guiar al usuario y hacerlo sentir bien en cada transacción que posea con el sistema.

3.13 Concepción de la ayuda.

La accesibilidad a la ayuda será en cada página de la aplicación y mostrando solo información de utilidad al usuario dependiendo de la pagina en que se encuentre. Cada página debe mostrar lo que se debe hacer y cómo hacerlo para que sirva de guía al usuario en cada operación en el sistema. Además de dar una explicación de los posibles mensajes de error del sistema en cualquier operación que se realice.

3.14 Conclusión

En este capítulo fueron expuestos diversos aspectos que describen como interacciona el sistema, fueron descritas las clases del análisis y las clases del diseño con estereotipos web. Por otro lado fueron detalladas cada una de las clases para que se conozcan las funcionalidades, y se describieron los diagramas de interacción. Por último fueron expuestos los principios de diseño seguidos para la implementación del sistema, entre los que se encuentran la interfaz de usuario, formato de salida de los reportes, tratamiento de errores, ayuda y seguridad.

Conclusiones

- Se analizaron los procesos de prueba que se realizan en el Proyecto de Calidad de la facultad 8 para el entendimiento de estos procesos.
- Se representó el negocio construyendo los modelos que a él tributan.
- Se identificaron las actividades automatizables para dar cumplimiento a los requisitos definidos por el cliente logrando una definición de las funcionalidades que abarquen dichos requisitos.
- Se generaron artefactos referentes al flujo de trabajo de análisis y artefactos del flujo de trabajo de diseño para que sirvan a la posterior implementación del sistema.

Recomendaciones

- Implementar el sistema para dar solución a los problemas existentes en el Grupo de Calidad de la Facultad 8.
- Lograr que el sistema se haga extensivo a otros grupos de Calidad en la Universidad luego de su implementación.
- Desarrollar un estudio sobre los métodos de prueba de caja blanca para incorporar funcionalidades asociadas a este tema.
- Incorporar funcionalidades asociadas a la clasificación de anomalías que luego de su posterior análisis permitan la aplicación de medidas correctivas, logrando de esta forma elevar la calidad de los procesos de prueba.

Referencias Bibliográficas

(Catalogo, 2007). Sistemas de Gestión de Calidad / ISO 9000. [En línea] 2007. http://www.catalogodesoftware.com/categoria.aspx?cid=124.

(EVA, 2007). Conferencia #2 El lenguaje XHTML. Principales etiquetas. *Conferencia #2 El lenguaje XHTML. Principales etiquetas.* [En línea] 2007. http://eva.uci.cu/file.php/266/Actividades_tema2/Conferencia_2_Xhtml_.pdf.

(EVA1, 2007). Conferencia #8 Introducción a las Hojas de estilo. Modelo de caja (Box Model). *Conferencia #8 Introducción a las Hojas de estilo. Modelo de caja (Box Model).* [En línea] 2007. http://eva.uci.cu/file.php/266/Actividades_Tema_3/Conferencia_8_Css_Modelo_de_caja.pdf.

(EVA2, 2008). Conferencia#10 Dhtml y JavaScript. [En línea] 2008. http://eva.uci.cu/file.php/266/Actividades_Tema_3/Conferencia_10_Dhtml_y_JavaScriptl_.pdf.

(EVA3, 2008). Conferencia_1. [En línea] 2008. http://eva.uci.cu/mod/resource/view.php?id=2241&subdir=/Conferencias_IS1_2007-2008.

(EVA4, 2008). Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 2008. http://eva.uci.cu/mod/resource/view.php?id=4599&subdir=/Materiales_Complementarios_Conf_7.

(EVA5, 2008). Fase de Inicio. Flujo de trabajo de requerimientos. [En línea] 2008. http://eva.uci.cu/mod/resource/view.php?id=12103.

(FREED, 2007). freedownloadmanager. [En línea] 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

(Peréz, 2007). esepe studio. [En línea] 2007. http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm

(PRESSMAN, 2005). *Ingeniería del Software, un enfoque práctico.* 2005.

(Cuesta, 2005). [En línea] 2005. http://www.latinspin.org/docs/sg200506.pdf.

Bibliografía

Biblioteca UCI [En línea] 2004. http://biblioteca.uci.cu/bives/titdigitales.htm#pro.

(RUP, 2003). *Ayuda de Rational Unified Process.* 2003.

(Hetzel, 1984). *The Complete Guide to Software Testing ,QED Information Sciences,Inc.,Wellesley ,MA.* 1984.

(Larman, 2004). UML y Patrones .Introducción al análisis y diseño orientado a objetos. 2004.

(KABIR, 2003). *La biblia de Servidor Apache 2. Española.* 2003.

Tendencias Ingenieria Software. [En línea] http://www.mitecnologico.com/Main/TendenciasIngenieriaSoftware.

Calidad del Software(II). [En línea] https://www.icai.es/contenidos/publicaciones/anales_get.php?id=842.

(Rational, 2007). [En línea] 2007. http://www.rational.com.ar/herramientas/rup.html.

Unified Modeling Language. [En línea] 2009. http://www.uml.org/.

ASP Fundamentos de programación en ASP 3.0. *ASP Fundamentos de programación en ASP 3.0.* [En línea] 2004. http://bibliodoc.uci.cu/pdf/reg01310.pdf.

Intellitech. [En línea] 2008. http://www.intellitech.net.mx/glosario.html.

Anexos

Anexo 1: Descripción de los casos de usos del sistema.

Tabla 42.- Descripción textual del caso de uso “Gestionar Lista de Chequeo”.

Caso de uso	
CU-2	Gestionar Lista de Chequeo
Propósito	Permite incluir, ver, modificar y eliminar la lista de chequeo.
Actores	Diseñador_CPLCH
Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Lista de Chequeo. El actor puede incluir, ver, modificar y eliminar una Lista de Chequeo. En caso de que seleccione la opción de incluir una Lista de Chequeo, el sistema da la posibilidad de introducir los datos necesarios para incluir una Lista de Chequeo. Si el actor elige la opción de ver una Lista de Chequeo, el sistema muestra el contenido de la Lista de Chequeo en cuestión. Si el actor elige la opción de modificar una Lista de Chequeo, el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Lista de Chequeo el sistema elimina la Lista de Chequeo en cuestión.

Tabla 43.- Descripción textual del caso de uso “Gestionar Diseño de Caso de Prueba”.

Caso de uso	
CU-3	Gestionar Diseño de Caso de Prueba
Propósito	Permite incluir, ver, modificar y eliminar un Diseño de Caso de Prueba.

Anexos

Actores	Diseñador_CPLCH
Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un Diseño de Caso de Prueba. El actor puede incluir, ver, modificar y eliminar un Diseño de Caso de Prueba. En caso de que seleccione la opción de incluir un Diseño de Caso de Prueba, el sistema da la posibilidad de introducir los datos necesarios para incluir un Diseño de Caso de Prueba. Si el actor elige la opción de ver un Diseño de Caso de Prueba, el sistema muestra el contenido del Diseño de Caso de Prueba en cuestión. Si el actor elige la opción de modificar un Diseño de Caso de Prueba, el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar un Diseño de Caso de Prueba el sistema elimina el Diseño de Caso de Prueba en cuestión.

Tabla 44.- Descripción textual del caso de uso “Gestionar Solicitud”.

Caso de uso	
CU-5	Gestionar Solicitud
Propósito	Permite incluir, ver, modificar y eliminar solicitudes
Actores	Lider_Proyecto,Jefe_Prueba

Anexos

Resumen:	El caso de uso inicia cuando el probador selecciona la opción que le permite realizar una acción sobre una Anomalía. El probador puede incluir, ver, modificar y eliminar una Anomalía. En caso de de que seleccione la opción de incluir una Anomalía, el sistema da la posibilidad de introducir los datos necesarios para incluir una Anomalía. Si el actor elige la opción de ver Anomalía el sistema muestra el contenido de dicha Anomalía. Si el actor selecciona la opción de modificar Anomalía. Si el actor selecciona la opción de eliminar una Anomalía el sistema elimina la anomalía en cuestión.

Tabla 45.- Descripción textual de caso de uso “Gestionar Respuesta Anomalía”.

Caso de uso	
CU-6	Gestionar Respuesta Anomalía
Propósito	Permite incluir, ver, modificar y eliminar respuesta de anomalías.
Actores	Solicitante_ED

Anexos

Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre una Respuesta de anomalía. El actor puede incluir, ver, modificar y eliminar una Respuesta de Anomalía. En caso de que seleccione la opción de incluir una Respuesta de Anomalía, el sistema da la posibilidad de introducir los datos necesarios para incluir una Respuesta de Anomalía. Si el actor elige la opción de ver una Respuesta de Anomalía el sistema muestra el contenido de la Respuesta de Anomalía en cuestión. Si el actor elige la opción de modificar una Respuesta de Anomalía., el sistema muestra los datos que pueden ser editables dentro de la misma, y una vez realizados los cambios, guardará las modificaciones. Si el actor elige la opción de eliminar una Respuesta de Anomalía el sistema elimina la Solicitud en cuestión.

Tabla 46.- Descripción textual del caso de uso “Evaluar Actividades Turno.

Caso de uso	
CU-8	Evaluar Actividades Turno
Propósito	Permite evaluar las actividades de los probadores finalizado un turno.
Actores	Jefe_Prueba
Resumen:	El caso de uso inicia cuando el actor selecciona la opción que le permite realizar la evaluación de la actividad finalizado un turno.

Tabla 47.- Descripción textual del caso de uso “Distribuir Trabajo”.

Caso de uso	
CU-8	Distribuir Trabajo

Anexos

Propósito	Permite distribuir el trabajo a los probadores.
Actores	Jefe_Prueba
Resumen:	El caso de uso inicia cuando el Jefe_Prueba accede a la sesión de trabajo para Distribución de Trabajo, selecciona la opción de Distribuir trabajo y realiza la distribución a los probadores existentes en el turno de trabajo.

Anexo 2: Diagramas de Clases del Diseño con estereotipos Web.

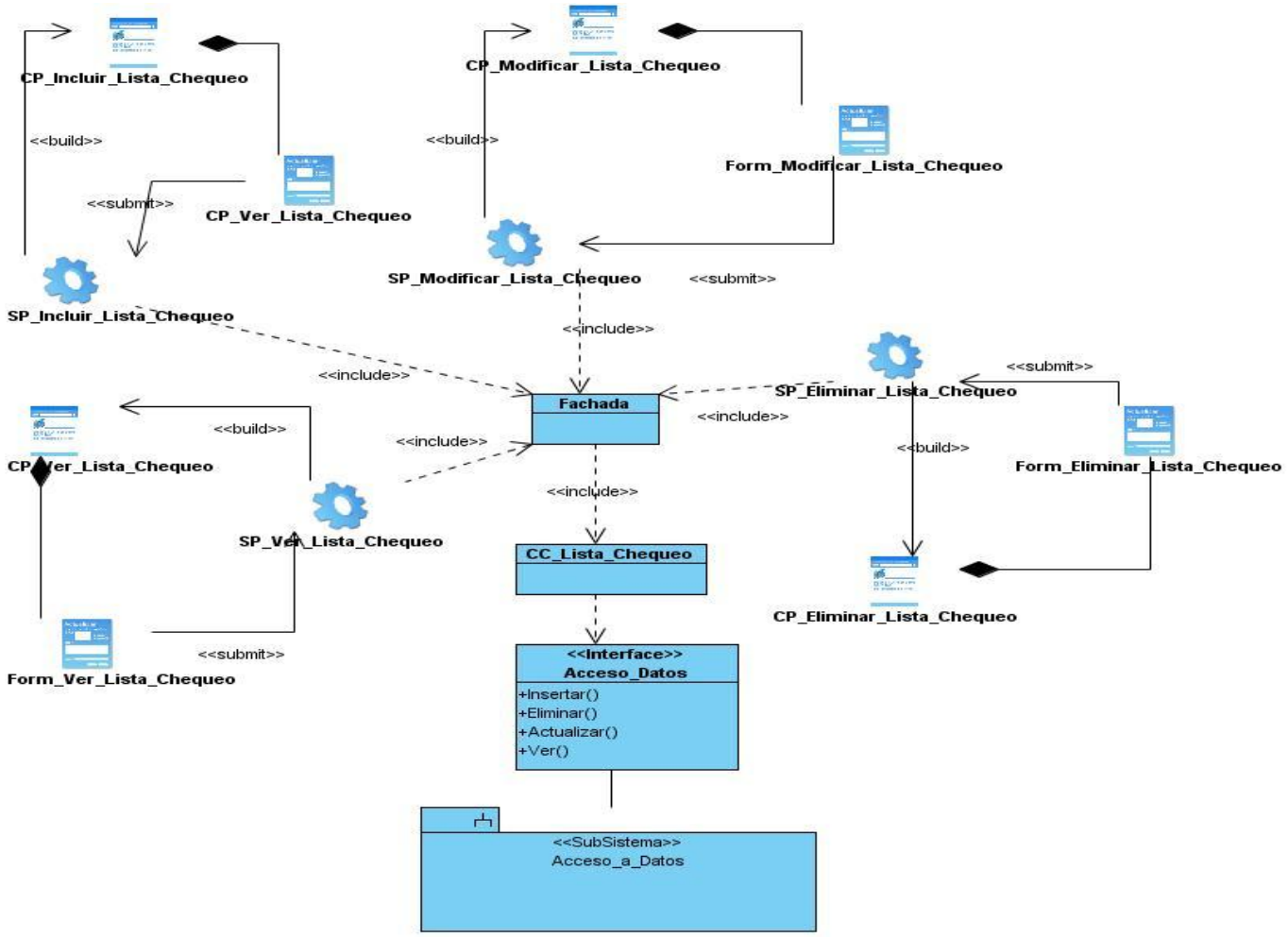


Figura 32.- Diagrama de clase del diseño del caso de uso “Gestionar Lista de Chequeo”.

Anexos

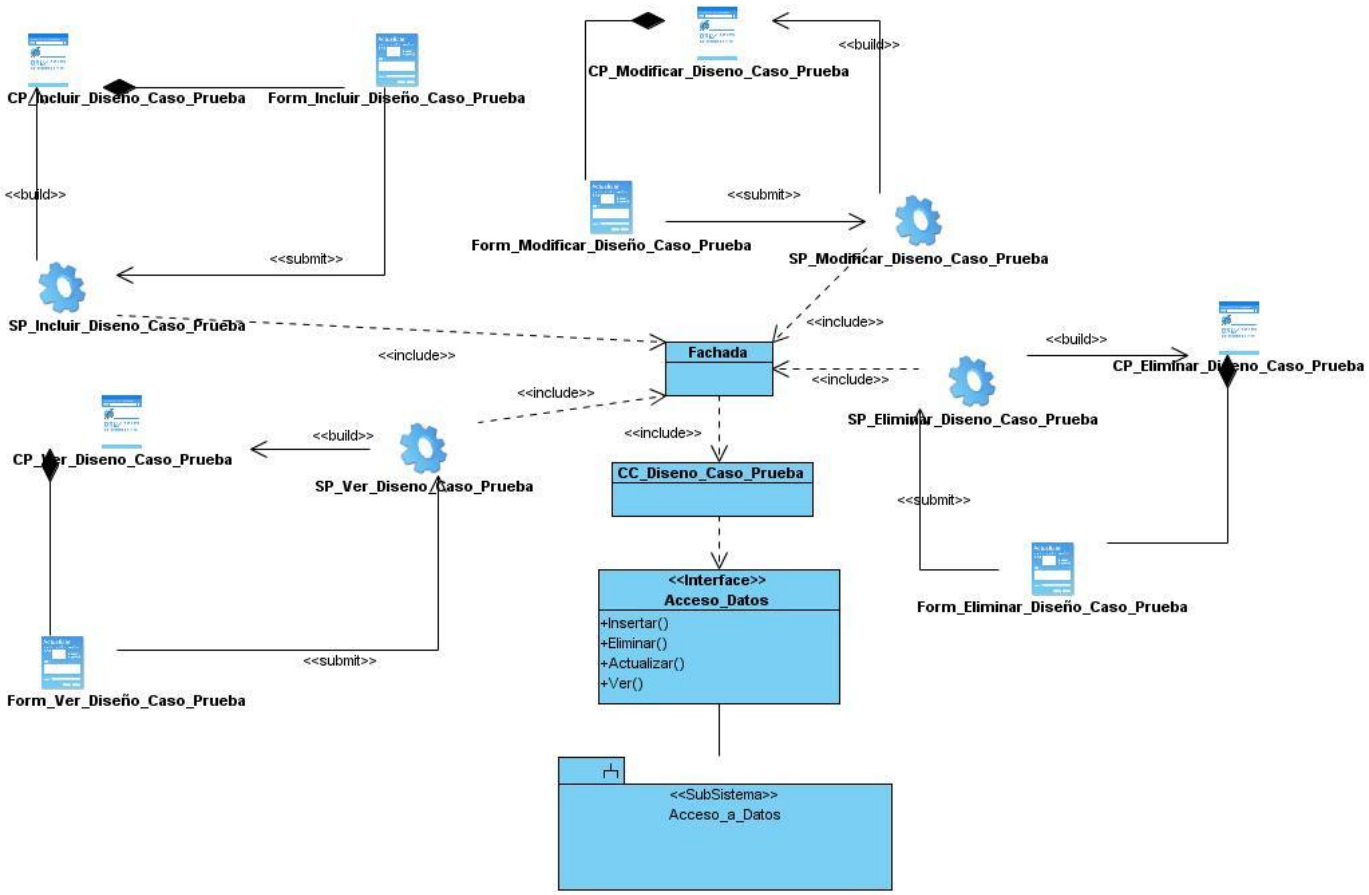


Figura 33.- Diagrama de clase del diseño del caso de uso “Gestionar Diseño de Caso de Prueba”.

Anexos

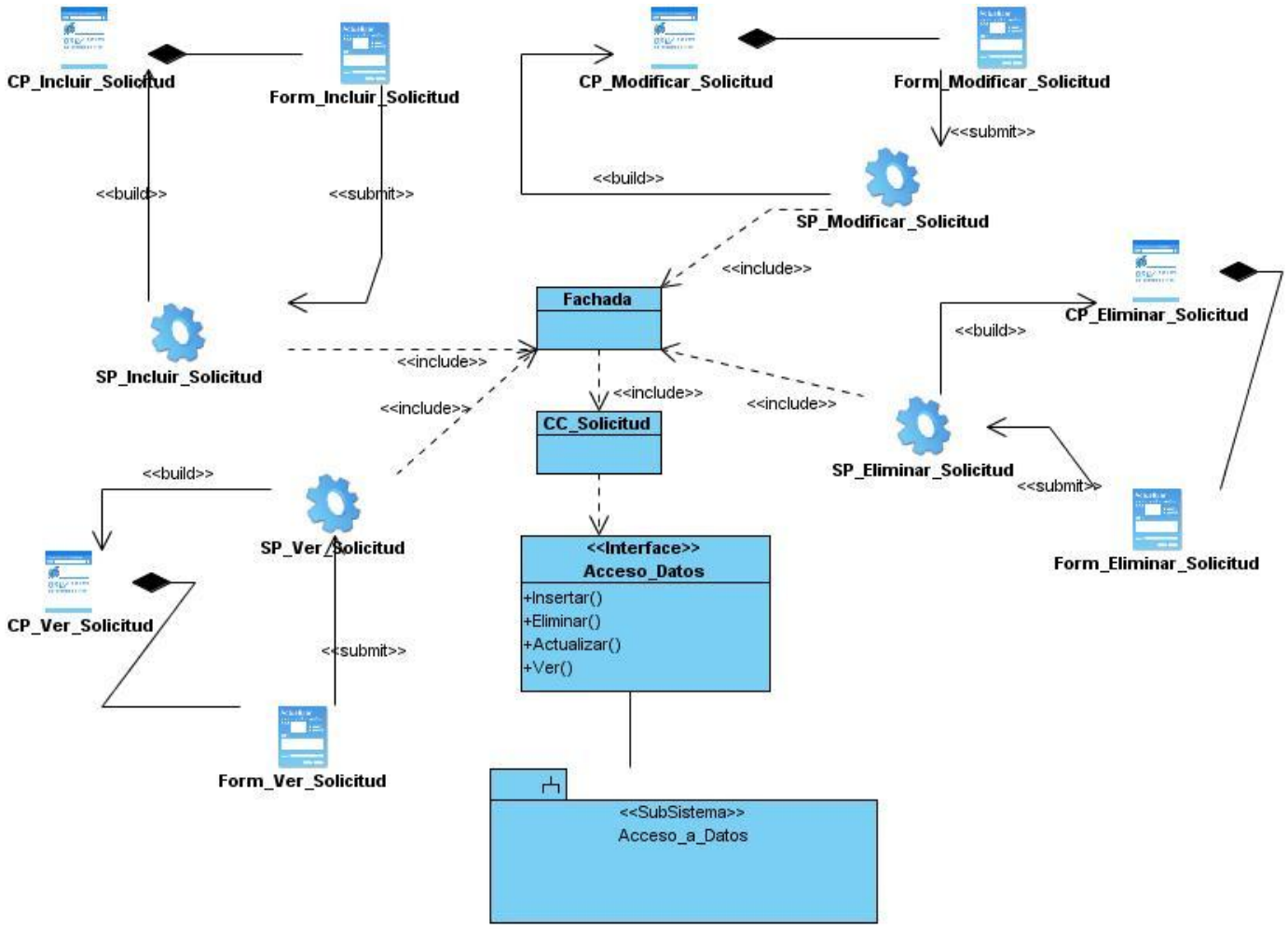


Figura 34.- Diagrama de clase del diseño del caso de uso “Gestionar Solicitud“.

Anexos

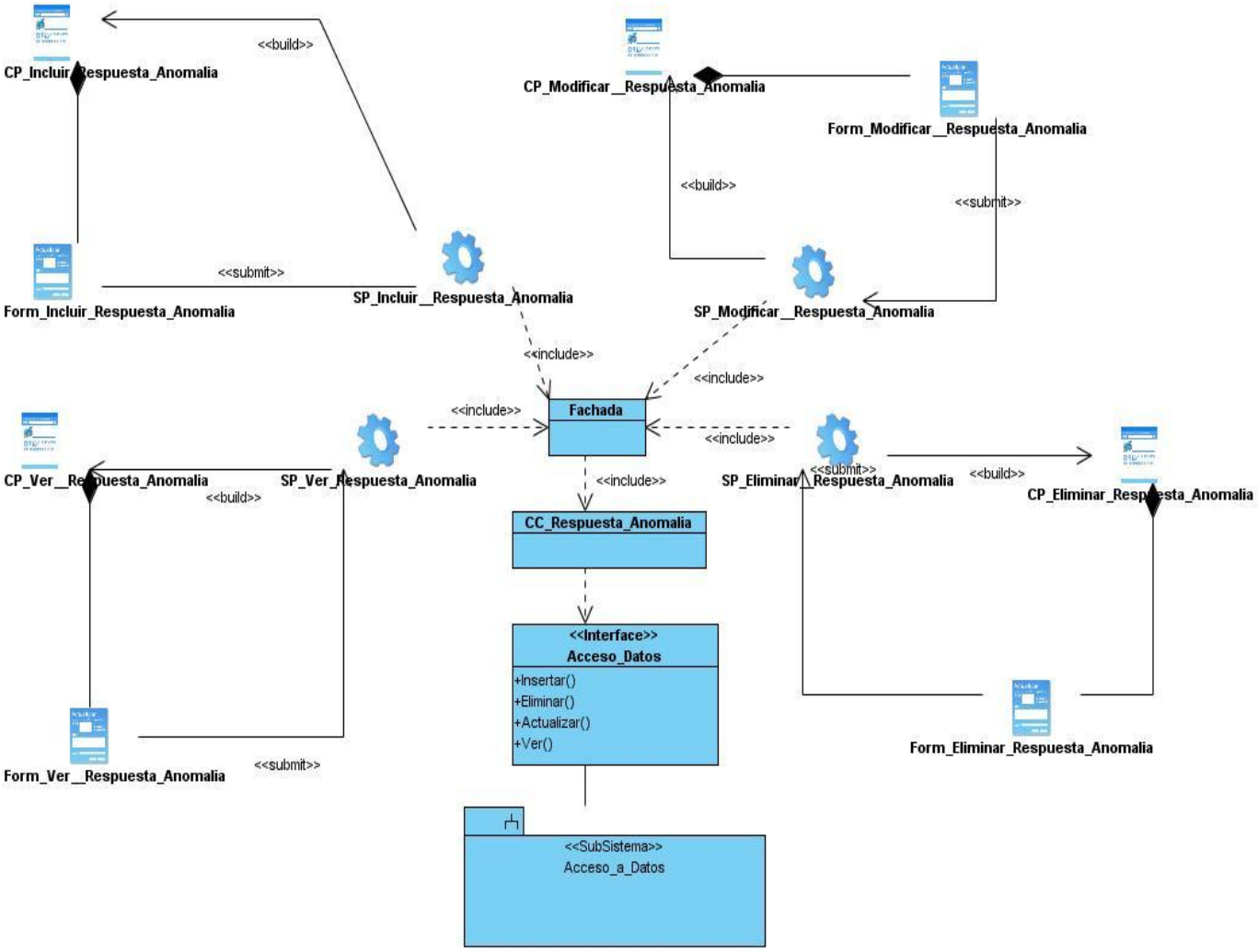


Figura 35.- Diagrama de clase del diseño del caso de uso “Gestionar Respuesta Anomalía”.

Anexos

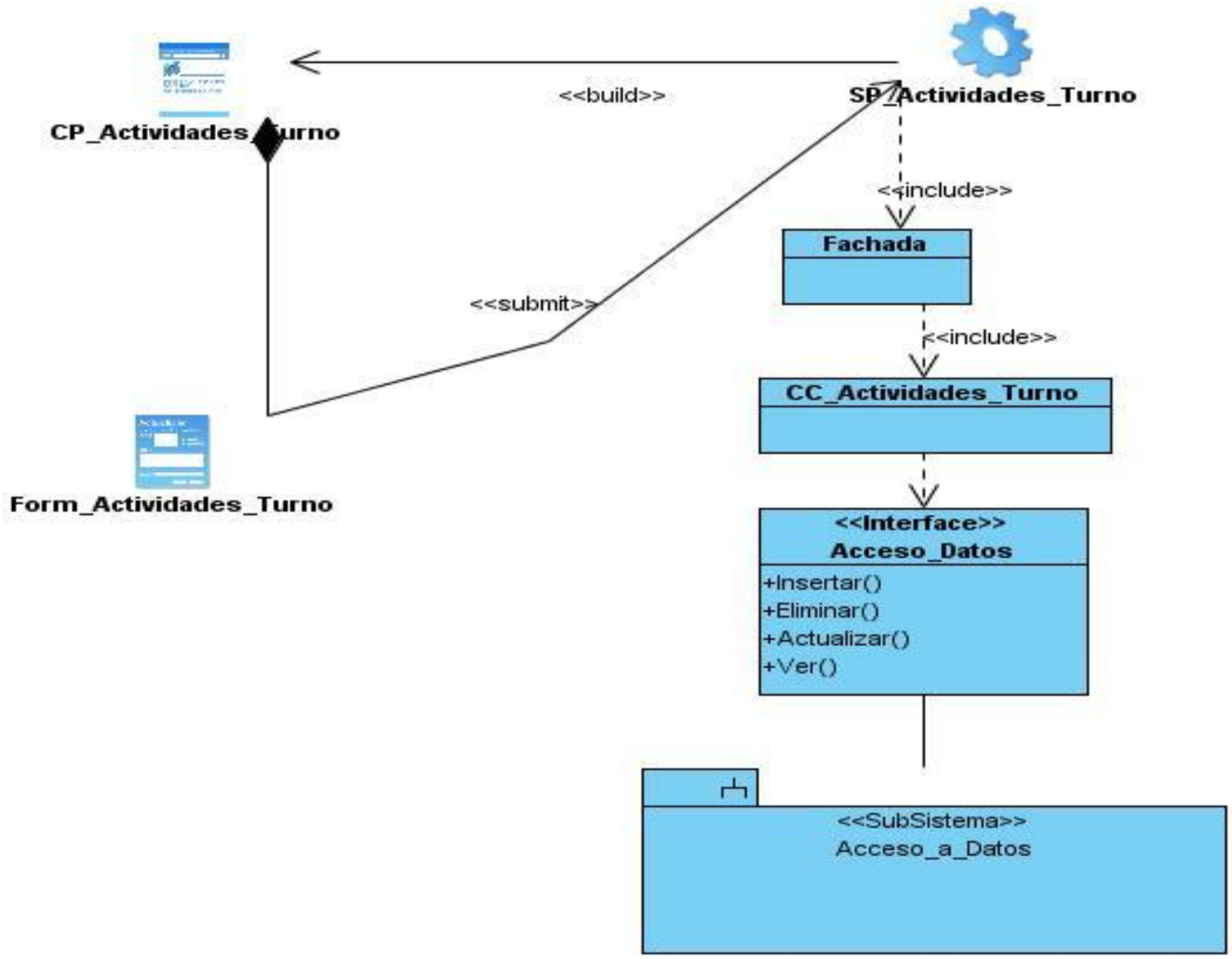


Figura 36.- Diagrama de clase del diseño del caso de uso “Evaluar Actividades Turno”.

Anexos

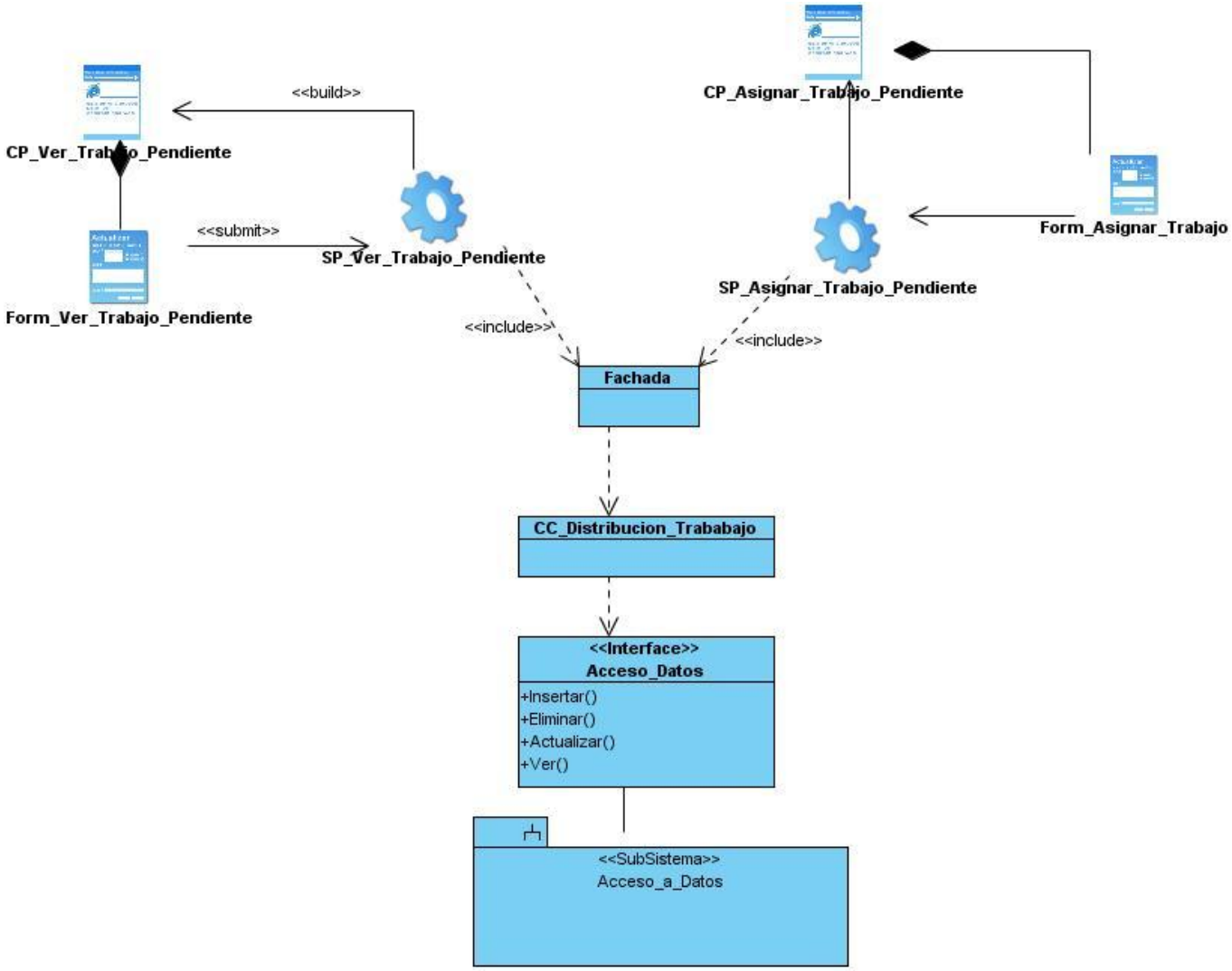


Figura 37.- Diagrama de clase del diseño del caso de uso “Distribuir Trabajo”.

Anexos

Anexo 3: Diagramas de interacción

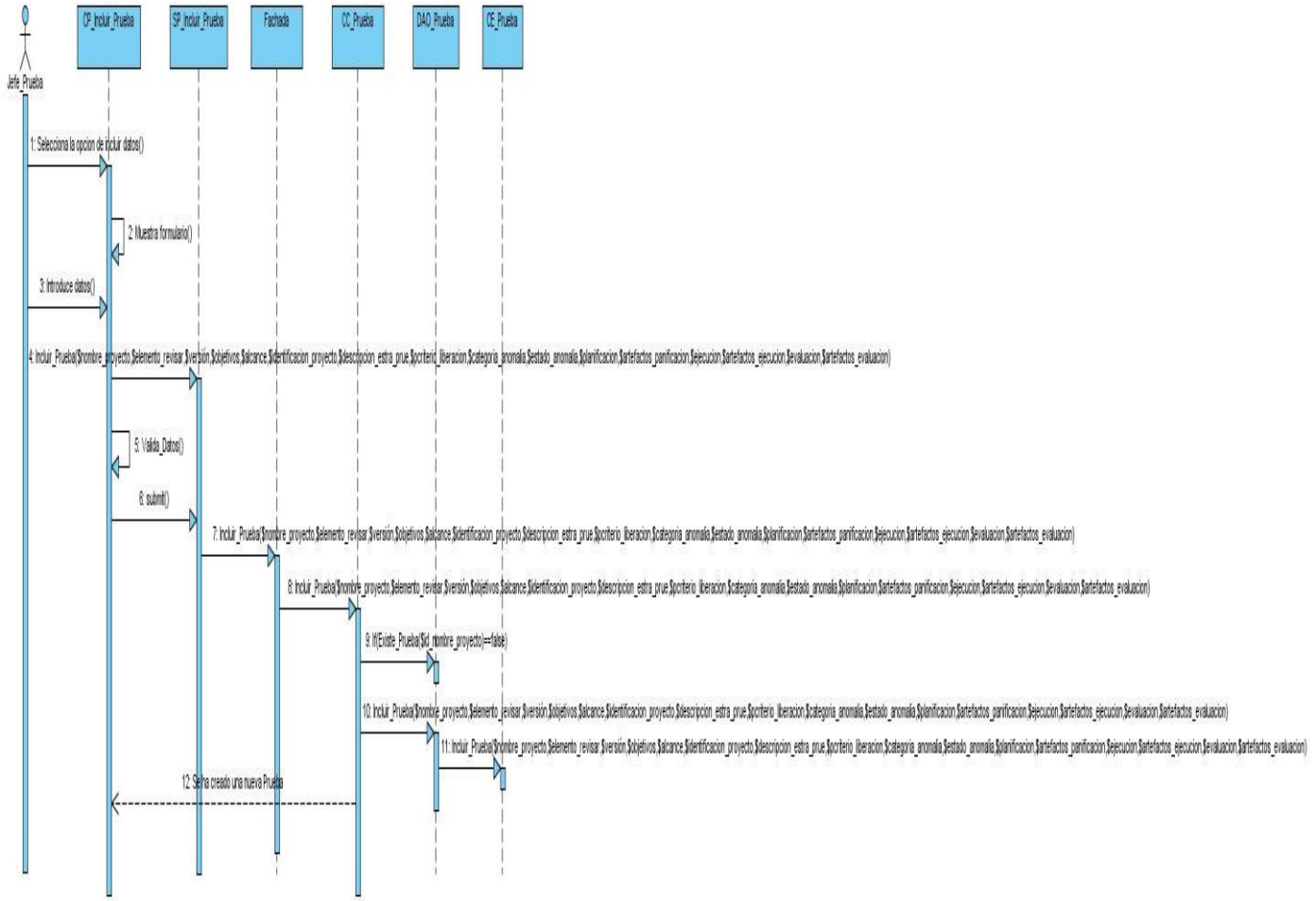


Figura 38.- Diagrama de secuencia del caso de uso “Gestionar Prueba”. “Escenario Incluir Prueba”.

Anexos

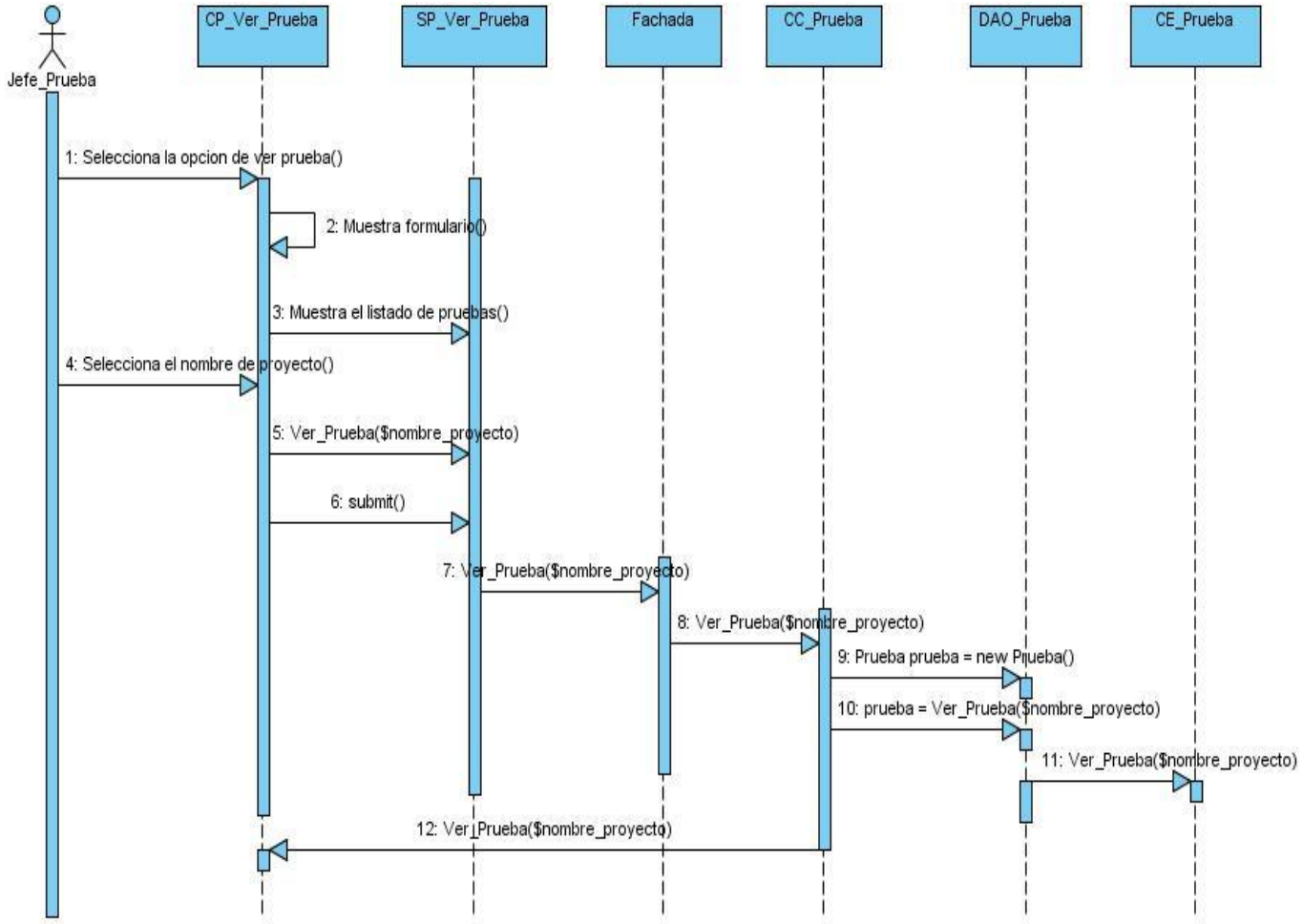


Figura 39.- Diagrama de secuencia del caso de uso “Gestionar Prueba”. “Escenario Ver Prueba”.

Anexos

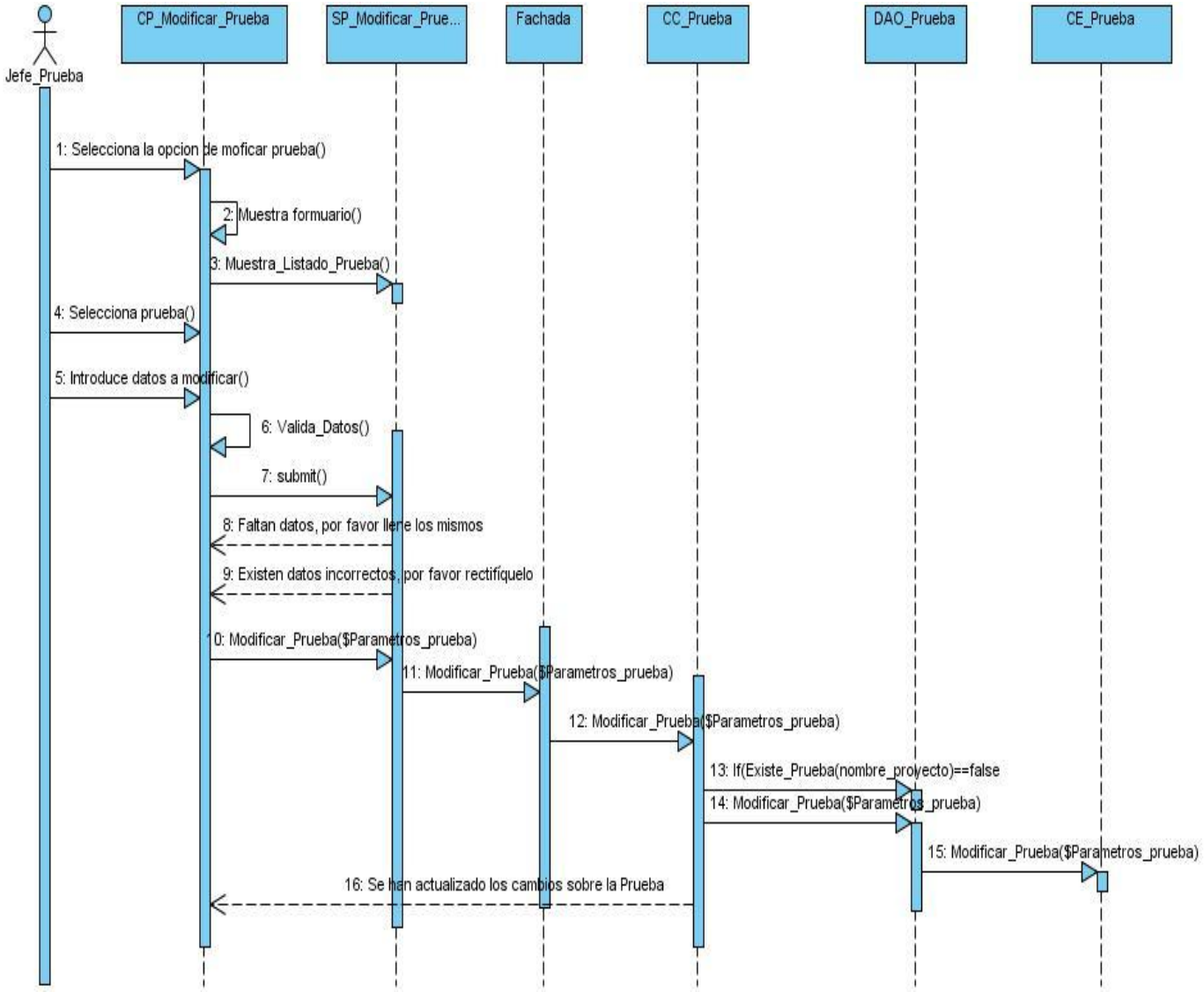


Figura 40.- Diagrama de secuencia del caso de uso “Gestionar Prueba". “Escenario Modificar Prueba".

Anexos

Anexo 4: Descripciones de las tablas.

Nombre	CE_Probador	
Tipo de clase	Entidad	
Atributo		**Tipo**
nombre		varchar
apellidos		varchar
usuario		varchar
contrasena		varchar
anno_carrrera		int
Responsabilidad		
Nombre	CEProbador(pnombre,papellidos,pusuario,pcontrasena, anno_carrrera)	
	GetNombre()	
	GetApellidos()	
	GetUsuario()	
	GetContrasena()	
	GetAnno_carrerera()	
Descripción	Clase entidad que representa la tabla Anomalía en la base de datos.	
Nombre	CE_Lider_Proyecto	
Tipo de clase	Entidad	
Atributo		**Tipo**
nombre		varchar
apellidos		varchar

Anexos

usuario	varchar
contrasena	varchar
anno_experiencia	int
Responsabilidad	
Nombre	CE_Lider_Proyecto(pnombre,papellidos,pusuario,pcontrasena, anno_experiencia)
	Getnombre()
	Getapellidos()
	Getusuario()
	GetContrasena()
	GetAnno_experiencia()
Descripción	Clase entidad que representa la tabla Líder_Proyecto en la base de datos.
Nombre	CE_Jefe_Prueba
Tipo de clase	Entidad

Atributo	**Tipo**
nombre	varchar
apellidos	varchar
usuario	varchar
contrasena	varchar
categoria	varchar
Responsabilidad	
Nombre	CE_Jefe_Prueba (pnombre,papellidos,pusuario,pcontrasena,pcategoria)
	GetNombre()

Anexos

	GetApellidos()
	GetUsuario()
	GetContrasena()
	GetCategoria()
Descripción	Clase entidad que representa la tabla Jefe_Proyecto en la base de datos.
Nombre	CE_Solicitante_ED
Tipo de clase	Entidad

Atributo	**Tipo**
nombre	varchar
apellidos	varchar
usuario	varchar
contrasena	varchar
rol_ED	varchar

Responsabilidad	
Nombre	CE_Solicitante_ED(pnombre,papellidos,pusuario,pcontrasena,rol_ED)
	GetNombre()
	GetApellidos()
	GetUsuario()
	GetContrasena()
	GetRol_ED()
Descripción	Clase entidad que representa la tabla Solicitante_ED en la base de datos.
Nombre	CE_Disenador_CPLCH

Anexos

Tipo de clase	Entidad	
Atributo		**Tipo**
nombre		varchar
apellidos		varchar
usuario		varchar
contrasena		varchar
grupo_cal		varchar
Responsabilidad		
Nombre	CE_Disenador_CPLCH(pnombre,papellidos,pusuario,pcontrasena,grupo_cal)	
	GetNombre()	
	GetApellidos()	
	GetUsuario()	
	GetContrasena()	
	GetGrupo_Cal()	
Descripción	Clase entidad que representa la tabla Disenador_CPLCH en la base de datos.	
Nombre	CE_ Persona	
Tipo de clase	Entidad	
Atributo		**Tipo**
usuario		varchar
contrasena		varchar
nombre		varchar
apellidos		varchar
Responsabilidad		

Anexos

Nombre	CE_Persona(pusuario,pcontrasenna,pnombre,papellidos)
	GetUsuario()
	GetContrasena()
	GetNombre()
	GetApellidos()
	SetUsuario(pUsuario)
	SetContrasena(pContrsenna)
	SetNombre(pNombre)
	SetaApellidos(pApellidos)
Descripción:	Clase entidad que representa la tabla Persona en la base de datos.
Nombre	CE_Solicitud
Tipo de clase	Entidad

Atributo	**Tipo**
Id_solicitud	int
nombre_solicitante	varchar
apellidos_solicitante	varchar
usuario_solicitante	varchar
contrasena_solicitante	varchar
Fecha_sol	Date

Responsabilidad	
Nombre	CE_Solicitud(p Id_solicitud,pnombre_solicitante, papellidos_solicitante,p usuario_solicitante,pcontrasena_solicitante, pfecha)
	GetId_solicitud()
	GetNombre_solicitante()

Anexos

	GetApellidos_solicitante()
	GetUsuario_solicitante()
	GetContrasena_solicitante()
	GetFecha_Sol()
Descripción	Clase entidad que representa la tabla Solicitud en la base de datos.
Nombre	CE_Prueba
Tipo de clase	Entidad

Atributo	**Tipo**
Id_prueba	varchar
nombre_proyecto_p	varchar
elemento_revisar	varchar
version	double
objetivos	varchar
alcance	varchar
identificacion_proyecto	varchar
descripcion_estra_prueba	varchar
criterio_liberacion	varchar
categoria_anomalia	varchar
estado_anomalia	varchar
planificacion	varchar
artefactos_planificacion	varchar
ejecucion	varchar
artefactos_ejecucion	varchar

Anexos

evaluacion	varchar
artefactos_evaluacion	varchar
Responsabilidad	
Nombre	CE_Prueba(pid_prueba,pnombre_proyecto_p,pelemento_revisar,p ersión,pobjetivos,palcance,pidentificacion_proyecto, pdescripcion_estra_prue,ppcriterio_liberacion, categoria_anomalia,pestado_anomalia,pplanificacion, partefactos_panificacion,pejecucion,partefactos_ejecucion, pevaluacion,partefactos_evaluacion)
	GetId_Prueba()
	GetNombre_proyecto_P()
	GetElemento_revisar()
	GetVersion()
	getObjetivos()
	GetAlcance()
	GetIdentificacion_proyecto()
	GetDescripcion_estra_prue()
	GetCriterio_liberacion()
	GetCategoria_anomalia()
	GetEstado_anomalia()
	GetPlanificación()
	GetArtefactos_panificacion()
	GetEjecucion()
	Getartefactos_ejecucion()
	GetEvaluación()

Anexos

	GetArtefactos_evaluacion()
	SetId_Prueba(pId_Prueba)
	SetNombre_proyecto_P(pNombre_proyecto_P)
	SetElemento_revisar(p)
	SetVersion(p)
	SetObjetivos(pObjetivos)
	SetAlcance(pAlcance)
	SetIdentificacion_proyecto(PIdentificacion_proyecto)
	SetDescripcion_estra_prue(PDescripcion_estra_prue)
	SetCriterio_liberacion(PCriterio_liberacion)
	SetCategoria_anomalia(pCategoria_anomalia)
	SetEstado_anomalia(pEstado_anomalia)
	SetPlanificacion(pPlanificacion)
	SetArtefactos_panificacion(pArtefactos_panificacion)
	SetEjecucion(pEjecucion)
	SetArtefactos_ejecucion(pArtefactos_ejecucion)
	SetEvaluacion(pEvaluacion)
	SetArtefactos_evaluacion(pArtefactos_evaluacion)
Descripción	Clase entidad que representa la tabla Prueba en la base de datos.
Nombre:	CE_Lista_Chequeo
Tipo de clase	Entidad

Atributo	**Tipo**

Anexos

Id_herramienta_prueba	varchar
nombre_proyecto	varchar
version	double
introduccion	varchar
proposito	varchar
objetivo	varchar
alcance	varchar
referencias	varchar
resumen	varchar
secciones	varchar
constancia_uso	varchar
Responsabilidad	
Nombre	CE_Lista_Chequeo(pid_herramienta_prueba,pnombre_proyecto,pversion, pintroduccion,pproposito ,pobjetivo,palcance,pobjetivo,palcance,preferencias,presumen,pconstancia_uso)
	GetId_herramienta_prueba()
	GetNombre_proyecto()
	GetVersion()
	GetIntroducción()
	GetProposito()
	GetObjetivo()
	GetAlcance()
	GetReferencias()
	GetResumen()

Anexos

	GetSecciones()
	GetConstancia_uso()
	SetId_Herramienta_Prueba(pid_herramienta_prueba)
	SetNombre_proyecto(pnombre_proyecto)
	SetVersion(pversion)
	SetIntroduccion(pIntroduccion)
	SetProposito(pProposito)
	SetObjetivo(pObjetivo)
	SetAlcance(pAlcance)
	SetReferencias(pReferencias)
	SetResumen(pResumen)
	SetSecciones(pSecciones)
	SetConstancia_uso(pConstancia_uso)
Descripción	Clase entidad que representa la tabla Lista_Chequeo en la base de datos.
Nombre	CE_Diseno_Caso_Prueba
Tipo de clase	Entidad

Atributo	**Tipo**
Id_herramienta_prueba	varchar
nombre_proyecto	varchar
version_modulo	double
version_proyecto	double
nombre_caso_uso	varchar
version_CP	double

Anexos

fecha	Date
descripcion	varchar
condiciones_ejecucion	varchar
autor	varchar
seccion_probar	varchar
seccion_revisar	varchar
Responsabilidad	
Nombre	CE_Diseno_Caso_Prueba(pid_herramienta_prueba,pnombre_proyecto,pversion_modulo, pversion_proyecto,pnombre_caso_uso,pversion_CP,pfecha,pversion, descripcion,pautor, pcondiciones_ejecucion,pseccion_probar,pseccion_revisar)
	GetId_herramienta_prueba()
	GetNombre_proyecto()
	GetVersion_modulo()
	GetVersion_proyecto(p)
	GetNombre_caso_uso()
	GetVersion_CP()
	GetFecha()
	GetDescripcion()
	GetAutor()
	GetCondiciones_ejecucion()
	GetSeccion_probar()
	GetSeccion_revisar()
	SetId_Herramienta_Prueba(pId_Herramienta_Prueba)

Anexos

	SetNombre_proyecto(pNombre_proyecto)
	SetVersion_modulo(pVersion_modulo)
	SetVersion_proyecto(pVersion_proyecto)
	SetNombre_caso_uso(pNombre_caso_uso)
	GetVersion_CP(pVersion_CP)
	SetFecha (pFecha)
	SetDescripcion(pDescripcion)
	SetAutor (pAutor)
	SetCondiciones_ejecucion (pCondiciones_ejecucion)
	SetSeccion_probar(pSeccion_probar)
	SetSeccion_revisar(pSeccion_revisar)
Descripción	Clase entidad que representa la tabla Diseno_Caso_Prueba en la base de datos.
Nombre	CE_Reporte
Tipo de clase	Entidad

Atributo	**Tipo**
id_reporte	int
tipo_reporte	varchar
fecha_reporte	Date

Responsabilidad	
Nombre	CE_Reporte(pid_reporte)
	GetId_Reporte()
	GetTipo_Reporte()
	GetFecha_Reporte()

Anexos

	SetId_Reporte(pId_Reporte)	
	SetTipo_Reporte(pTipo_Reporte)	
	SetFecha_Reporte(pFecha_Reporte)	
Descripción:	Clase entidad que representa la tabla Reporte en la base de datos.	
Nombre	CE_Reporte_PPP	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_reporte		int
cantidad_proyectos		int
listado_proyectos		CE_Proyecto
Responsabilidad		
Nombre	CE_Reporte_PPP(pid_reporte)	
	id_reporte	
	Add_Proyecto(CE_Proyecto)	
	Obtener_Proyecto(int)	
	GetCantidad_Proyectos()	
	Get_Listado_Proyectos()	
Descripción:	Clase entidad que representa la tabla Reporte_PPP en la base de datos.	
Nombre	CE_Reporte	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_reporte		int
cantidad_dias_trabajados		int
cantidad_anomalias		int

Anexos

listado_anomalias		CE_Anomalia
Responsabilidad		
Nombre	CE_Reporte_PEP(pid_reporte)	
	GetCantidad_dias_trabajados()	
	Get_Cantidad_anomalias()	
	Add_Anomalia(Anomalia)	
	Obtener_Anomalia(int)	
	Get_Listado_Anomalias()	
Descripción:	Clase entidad que representa la tabla Reporte_PEP en la base de datos.	
Nombre	CE_Reporte_GIARF	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_reporte		int
fecha_giarf		Date
listado_anomalias		CE_Anomalia
Responsabilidad		
Nombre	CE_Reporte_GIARF(pid_reporte)	
	GetFecha_Giarf()	
	Add_Anomalia(CE_Anomalia)	
	Obtener_Anomalia(int)	
	GetListado_Anomalias()	
Descripción:	Clase entidad que representa la tabla Reporte_ GIARF en la base de datos.	

Anexos

Nombre	CE_Herramienta_Prueba	
Tipo de clase	Entidad	
Atributo		**Tipo**
Id_herramienta_prueba		varchar
nombre_proyecto		varchar
Responsabilidad		
Nombre	CE_Herramienta_Prueba(pid_herramienta_prueba,pnombre_proyecto)	
	GetId_Herramienta_Prueba()	
	GetNombreProyecto()	
	SetId_Herramienta_Prueba(pId_Herramienta_Prueba)	
	SetNombreProyecto(pNombreProyecto)	
Descripción	Clase entidad que representa la tabla Herramienta_Prueba en la base de datos.	
Nombre	CE_Turno	
Tipo de clase	Entidad	
Atributo		**Tipo**
fecha		Date
sesion		varchar
Responsabilidad		
Nombre	CE_Turno(pfecha,psesion)	
	GetFecha()	
	GetSesion()	
	SetFecha(pFecha)	
	SetSesion(pSesion)	
Descripción	Clase entidad que representa la tabla Turno en la base de datos.	

Anexos

Nombre	CE_Actividades_Turno	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_actividades_turno		varchar
turno		Turno
evaluacion		varchar
Responsabilidad		
Nombre	CE_ Actividades_Turno (pId_actividades_turno,pturno ,pevaluacion)	
	Get Id_actividades_turno ()	
	GetTurno()	
	GetEvaluacion()	
Descripción	Clase entidad que representa la tabla Actividades_Turno en la base de datos.	
Nombre	CE_ Distribucion_Trabajo	
Tipo de clase	Entidad	
Atributo		**Tipo**
turno		Turno
lista_probadores		CE_Probador
Responsabilidad		
Nombre	CE_Distribucion_Trabajo(pturno,plista_probadores)	
	GetTurno()	
	Add_Probador(CE_Probador)	
	ObtenerProbador(int)	
	GetLista_probadores()	
Descripción	Clase entidad que representa la tabla Distribucion_Trabajo en la base de datos.	

Anexos

Nombre	CE_Iteracion	
Tipo de clase	Entidad	
Atributo		**Tipo**
Id_iteracion		int
lista_probadores		CE_Probador
lista_anomalias		CE_Anomalia
Responsabilidad		
Nombre	CE_Iteracion (pid_iteracion)	
	GetIdIteracion()	
	GetLista_Probadores()	
	GetLista_Anomalias()	
	Add_Probador(CE_Probador)	
	Add_Anomalia(CE_Anomalia)	
	Obtener_Probador(int)	
	Obtener_Anomalia(int)	
	SetIdIteracion(pIteracion)	
Descripción	Clase entidad que representa la tabla Iteracion en la base de datos.	
Nombre	CE_Respuesta_Anomalia	
Tipo de clase	**Entidad**	
Atributo		**Tipo**
Id_respueta		int
nombre_redactor		varchar
usuario_redactor		varchar
contrasenna_redactor		varchar

Anexos

descripción_RA	varchar
Responsabilidad	
Nombre	CE_Respuesta_Anomalia(respueta, nombre_redactor, usuario_redactor ,contrasenna_redactor, descripción_RA)
	GetId_respueta()
	GetNombre_redactor()
	GetUsuario_redactor()
	GetContrasenna_redactor()
	GetDescripción_RA()
	SetId_respueta(pId_respueta)
	SetNombre_redactor(pNombre_redactor)
	SetUsuario_redactor(pUsuario_redactor)
	SetContrasenna_redactor(pContrasenna_redactor)
	SetDescripción_RA(pDescripción_RA)
Descripción	
Nombre	CE_Escenario
Tipo de clase	Entidad

Atributo	**Tipo**
id_escenario	int
nombre_escenario	varchar
listado_variables	tipo_Variable
respuesta_sistema	varchar
resultado_prueba	varchar
Responsabilidad	
Nombre	CE_Escenario(id_escenario)

Anexos

	GetIdEscenario()
	GetNombre_escenario()
	GetListado_variables()
	GetRespuesta_Sistema()
	GetResultado_Prueba()
	SetNombre_Escenario(pNombre_Escenario)
	SetRespuesta_Sistema(pRespuesta_Sistema)
	SetResultado_Prueba(pResultado_Prueba)
	Add_Variables(variable)
	ObtenerVariable(int)
Descripción	Clase entidad que representa la tabla Escenario en la base de datos.
Nombre	CE_Seccion_Parte
Tipo de clase	Entidad

Atributo	**Tipo**
id_seccion_parte	int
nombre_seccion	varchar
listado_esc_desc_fluj	CE_ Esc_Desc_Fluj

Responsabilidad	
Nombre	CE_Seccion_Parte (id_seccion_parte)
	GetId_seccion_parte()
	GetNombre_seccion()
	GetListado_esc_desc_fluj()
	SetId_seccion_parte(pId_seccion_parte)
	SetNombre_seccion(pNombre_seccion)

Anexos

	AddEsc_Desc_Fluj(esc_desc_fluj)	
	ObtenerEsc_Desc_Fluj(int)	
Descripción	Clase entidad que representa la tabla Seccion_Parte en la base de datos.	
Nombre	CE_ Seccion_Probar	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_seccion_probar		int
listado_seccion_parte		CE_Seccion_Parte
Responsabilidad		
Nombre	CE_ Seccion_Probar (id_seccion_probar)	
	GetId_seccion_probar()	
	GetListado_seccion_parte()	
	SetId_Seccion_Probar(id_seccion_probar)	
	AddSeccion_Parte(seccion_parte)	
	ObtenerSeccion_Parte(int)	
Descripción	Clase entidad que representa la tabla Seccion_Probar en la base de datos.	
Nombre	CE_Seccion_Revisar	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_seccion_revisar		int
listado_escenario		CE_Escenario
Responsabilidad		
Nombre	CE_ Seccion_Probar (id_seccion_revisar)	
	GetId_seccion_revisar()	

Anexos

	GetListado_Escenario()
	SetId_Seccion_Parte(id_seccion_revisar)
	AddSeccion_Escenario(escenario)
	ObtenerEscenario(int)
Descripción	Clase entidad que representa la tabla Seccion_Revisar en la base de datos.
Nombre	CE_Esc_Desc_Fluj
Tipo de clase	Entidad

Atributo	**Tipo**
id_esc_des_fluj	int
escenario	varchar
descripcion	varchar
flujo	varchar

Responsabilidad	
Nombre	CE_ Seccion_Probar (id_esc_desc_fluj)
	GetId_Esc_Desc_Fluj()
	GetScenario()
	GetDescripcion()
	GetFlujo(flujo)
	SetId_Esc_Des_Fluj(esc_desc_fluj)
	SetEscenario(escenario)
	SetDescripcion(descripcion)
	SetFlujo(flujo)
Descripción	Clase entidad que representa la tabla Esc_Desc_Fluj en la base de datos.

Anexos

nombre	CE_Seccion_Revisar	
Tipo de clase	Entidad	
Atributo		**Tipo**
id_tipo_variable		varchar
variable		varchar
Responsabilidad		
Nombre	CE_Tipo_Variable(id_tipo_variable)	
	GetId_tipo_variable()	
	GetVariable()	
	SetId_tipo_variable(id_tipo_variable)	
	SetVariable(variable)	
Descripción	Clase entidad que representa la tabla Tipo_Variable en la base de datos.	
Nombre	CC_Prueba	
Tipo de clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion, categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion, artefactos_evaluacion);	
	Existe_Prueba(nombre_proyecto)	
	Existe_Prueba(id_prueba)	
	Ver_Prueba(nombre_proyecto)	
	Modificar_Prueba(nombre_proyecto, elemento_revisar, version,objetivos,alcance, identificacion_proyecto, descripcion_estra_prueba, criterio_liberacion,	

Anexos

	categoria_anomalia, estado_anomalia, planificacion, artefactos_panificacion, artefactos_planificacion, ejecucion, artefactos_ejecucion, evaluacion, artefactos_evaluacion)
	Eliminar_Prueba(nombre_proyecto)
Descripción	Controla las pruebas que se gestionan en el sistema por parte del jefe de proyecto.

Nombre	CC_Respuesta_Anomalia	
Tipo de clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_ redactor,contraseña_redactor,descripción_RA);	
	Existe_Respuesta_Anomalia(id_anomalia)	
	Cantidad_Respuestas_Anomalias_Dado_Nombre_Proyecto (nombre_proyecto)	
	Ver_Respuesta_Anomalia(Id_respuesta)	
	Modificar_Respuesta_Anomalia(Id_respuesta,nombre_redactor,usuario_ redactor,contraseña_redactor,descripción_RA)	
	Eliminar_Respuesta_Anomalia(id_anomalia)	
Descripción	Controla las respuestas de que puede realizar el solicitante del equipo de desarrollo.	
Nombre	CC_Lista_Chequeo	
Tipo de clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
Nombre	Incluir_Lista_Chequeo(id_herramienta_prueba,nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)	

Anexos

	Ver_Lista_Chequeo(id_herramienta_prueba)
	Modificar_Lista_Chequeo(id_herramienta_prueba,nombre_proyecto , version, introduccion, proposito, objetivo, alcance, referencias, resumen, constancia_uso)
	Buscar_Lista_Chequeo(id_herramienta_prueba)
	Eliminar_Lista_Chequeo(id_herramienta_prueba)
	Existe_ Lista_Chequeo(id_herramienta_prueba)
Descripción	Controla las listas de chequeo de los proyectos en prueba.
Nombre	CC_Diseno_Caso_Prueba
Tipo de clase	Controladora

Atributo	**Tipo**

Responsabilidad	
Nombre	Incluir_ Diseno_Caso_Prueba (id_herramienta_prueba ,nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)
	Ver_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Modificar_Diseno_Caso_Prueba (nombre_proyecto , version_modulo, version_proyecto, nombre_caso_uso, version_CP, fecha, version, descripcion, autor, seccion_probar, seccion_revisar)
	Buscar_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Eliminar_ Diseno_Caso_Prueba (id_herramienta_prueba)
	Existe__ Diseno_Caso_Prueba(id_herramienta_prueba)
Descripción	Controla los diseños de caso de prueba de los proyectos en prueba.
Nombre	CC_Reporte
Tipo de clase	Controladora

Atributo	**Tipo**

Anexos

Responsabilidad	
Nombre	Ver_ Reporte_Probador_por_Proyectos(usuario)
	Cantidad_Proyectos_Probados(usuario)
	Ver_ Reporte_Probador_en_Proyectos(usuario,nombre_proyecto)
	Buscar_Anomalias_Probador(nombre_proyecto,usuario)
	Buscar_Anomalias_Probador_por_Rango_Fecha(nombre_proyecto,usuario,rango_fecha)
	Cantidad_anomalias(nombre_proyecto,usuario,rango_fecha)
	Cantidad_Dias_Trabajados(nombre_proyecto,usuario,rango_fecha)
	Generar_Informe_Anomalia(usuario,rango_fecha)
Descripción	Controla los reportes que se pueden hacer en el sistema por parte de jefe de prueba.
Nombre	CC_Activiades_Turno
Tipo de clase	Controladora
Atributo	**Tipo**
Responsabilidad	
Nombre	Buscar_Probador_Por_Turno(usuario)
	Listado_Probador_Por_Turno(turno)
	Evaluar_Probador(turno)
	Dar_Evaluacion(evaluacion,usuario)
Descripción	Controla los reportes que se pueden hacer en el sistema por parte de jefe de prueba.
Nombre	CC_Solicitud
Tipo de clase	Controladora
Atributo	**Tipo**

Anexos

Responsabilidad	
Nombre	Incluir_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contrasenna_solicitante, nombre_proyecto, fecha_solicitud);
	Existe_Solicitud(id_solicitud)
	Obtener_Solicitanter(id_solicitud)
	Ver_Solicitud(id_solicitud)
	Modificar_Solicitud(Id_solicitud,nombre_solicitante,usuario_solicitante, contrasena_solicitante, nombre_proyecto, fecha_solicitud)
	Eliminar_Solicitud(id_solicitud)
	Existe_Solicitud(id_solicitud)
Descripción	Controla las solicitudes que se gestionan en el sistema por parte del solicitante del equipo de desarrollo.
Nombre	CC_Distribucion_Trabajo
Tipo de clase	Controladora
Atributo	**Tipo**
Responsabilidad	
Nombre	Obtener_Distribucion(id_distribucion)
	Ver_Trabajo_Pendiente()
	Asignar_Trabajo_Pendiente()
	Buscar_Trabajo()
	Mostrar_Iteracion_Prueba()
	Listar_CasoUso_Probados()
	Listar_CasoUso_No_Probados()
Descripción	Controla las solicitudes que se gestionan en el sistema por parte del solicitante del

Anexos

	equipo de desarrollo.	
Nombre	Base_Datos	
Tipo de clase	DAO	
Atributo		**Tipo**
servidor		varchar
usuario		varchar
password		varchar
basedatos		varchar
conexion		varchar
error		varchar
AdoDB		ADONewConnection
Responsabilidad		
Nombre	Base_Datos(s=SERVIDOR, u=USUARIO, p=PASSWORD, bd= BASEDATOS, mt_bd=MOTOR_BD)	
	Abrir_Conexion ()	
	Cerrar_Conexion ()	
	Incluir (sql)	
	Modificar (sql)	
	Eliminar (sql)	
	Ver(sql)	
	Ejecutar_Consulta(sql)	
	Ejecutar_Insercion(sql)	
	Cargar_Registro (sql)	
	Empezar_Transaccion ()	
	Terminar_Transaccion ()	
	Ocurrio_Error_Transaccion ()	
	Forzar_Transaccion ()	
Descripción	Clase encargada de realizar la conexión a la base de datos y ejecutar las consultas, haciendo uso del paquete ADOdb.	

Glosario de Términos

GTC: Gestión total de calidad

Prueba Caja Negra: Son técnicas de diseño de casos de prueba que permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

Prueba Caja Blanca: Son técnicas de diseño de casos de prueba que usa la estructura del control del diseño procedimental para obtener los casos de prueba.

Licencia BSD: Licencia de software otorgada principalmente para los sistemas BSD. Pertenece al grupo de licencias de Software Libre; tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

HTML: Lenguaje de etiqueta estandarizado para la creación de documentos para la web.

CUN: Caso de uso del negocio.

BD: Base de Datos

RUP: Rational Unified Process (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.

Plantilla_DCP: Plantilla de Diseño de Caso de Prueba.

Diseñador_CPLCH: Responsable por el equipo de prueba de la realización de Diseños de Caso de Prueba y Listas de Chuequeo.

Solicitante_ED: Responsable por el equipo de desarrollo de responder las anomalías detectadas por el equipo de prueba.

Representante_ED: Representante del equipo de desarrollo.

Representante_EP: Representante del equipo de prueba.

Lider_Proyecto: Jefe del proyecto en prueba.

Probador: Reponsable del equipo de prueba de la realización de las pruebas.

Glosario de Términos

Jefe_Prueba: Encargado del proyecto de calidad de hacer cumplir las pruebas y realizar reportes.

SGBD: Sistema de Gestión de Bases de Datos. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios.

SQL: Es un lenguaje declarativo de acceso a bases de datos que permite especificar diversos tipos de operaciones sobre las mismas. Permite lanzar consultas con el fin de recuperar información de interés de una base de datos.

DAO: Patrón de Objeto de Acceso a Datos, consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos.

MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras.

CSS: Tecnología que permite crear páginas web con un diseño más exacto, añadiendo mayores posibilidades a HTML y permitiendo una mayor separación entre la información y la presentación.

API: Especificación de una librería o utilidad que documenta su interfaz y permite su uso sin conocimiento de su interior.

Framework: Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

HTTP: Protocolo de nivel de aplicación usado extensivamente en Internet para el acceso a documentos.

ISO: Federación de alcance mundial integrada por cuerpos de estandarización nacionales de gran número de países, que promueve el desarrollo de la estandarización.

Java : Plataforma para el desarrollo de software creada por Sun Microsystems, ampliamente extendida hoy en día, que otorga independencia de plataforma al software creado en ella y lo provee de una gran cantidad de APIs estandarizados.

MVC: (Modelo Vista Controlador) Patrón arquitectónico desarrollado para interfaces gráficas que resalta la importancia de una separación clara entre la presentación de datos y la lógica de negocio de una aplicación.

UML: Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

XP: Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación y coraje.